Technische Universität München
Fakultät für Physik

Master's thesis

Towards automating abundance tomography for type Ia supernovae

using machine learning techniques

by Stefan Lietzau

April 28, 2017

Contents

Abstract

In this thesis we present techniques to automate the fitting of spectral time series of Type Ia supernovae (SNe Ia). These transient objects play an important part in the chemical evolution of the Universe and are used as important distance indicators to map out the expansion history of the Universe. Despite their importance, many aspects of these transient events are still unknown. An important tool to study the explosion in detail is the abundance tomography method in which a model is fitted to the spectral time series of an observation using a fast spectral synthesis code.

In this work we present first steps to accelerate and automate this method in an attempt to explore the parameter space surrounding the best fitting set of parameters and to highlight degeneracies. To this end, we develop a framework that uses *Machine Learning* (ML) techniques to quickly generate spectra and then use Markov chain Monte Carlo (MCMC) methods to explore the parameter space with the help of Bayesian statistics to construct a measure of the quality of fit. After extensively testing and validating the framework, we apply it to supernova (SN) 2002bo, which was already the subject of similar studies done manually. However, reproducing the results found in literature is a challenge that needs further research beyond this thesis.

1 Background and previous work

Supernovae (SNe) are among brightest astronomical objects in the Universe which can even outshine their entire host galaxy during their evolution. A subclass, the SNe Ia have played a crucial role in recent years in the field of cosmology due to their use as standardizable candles. An empirical relation, between the width of a lightcurve and the intrinsic luminosity of a SN Ia has been identified (Phillips, 1993; Pskovskii, 1984). Using this relation SN Ia were used as distance indicators, most famously by Riess et al. (1998) and Perlmutter et al. (1999). They established the accelerated expansion of the Universe which was awarded with the Nobel prize in 2011. Besides their importance in cosmology, SN Ia play an important role in other disciplines of astrophysical research. For example in galactic chemical evolution because they are the main source of stable iron (see Matteucci & Greggio, 1986; Kobayashi et al., 1998). Moreover, they are discussed in the context of triggered star formation (see Leibundgut, 2000).

However, their importance is contrasted by persisting uncertainties about the nature of these objects and the physical processes that govern them (see Hillebrandt et al., 2013). The consensus in the astrophysical community is that SN Ia are thermonuclear explosions and full disruptions of a carbon/oxygen white dwarf (CO-WD) (Hoyle & Fowler, 1960; Truran et al., 1967). This thermonuclear burning produces large quantities of radioactive ^{56}Ni, whose decay through ^{56}Co to ^{56}Fe powers the intense light output of these systems which we observe (see Pankey, 1962; Truran et al., 1967; Colgate & McKee, 1969; Kuchner et al., 1994). However, the exact nature of the progenitor system and the details of the explosion are still subject of active debate and ongoing research.

1.1 Explosion physics

The evolution of low mass stars ($< 8\,M_{\mathrm{ZAMS}}$) ends in a compact white dwarf (WD) which has exhausted its nuclear burning fuel and thus, no burning takes place (see Kippenhahn et al., 2012). These dense objects are gravitationally stabilized through the pressure of the degenerate electron gas (see Shapiro & Teukolsky, 1983) and their only evolution consists of gradually cooling down. Thus, to trigger an explosion an external influence is required which comes in the form of a companion star (see *e.g.,* Hillebrandt & Niemeyer, 2000; Hillebrandt et al., 2013).

Historically two possible realizations of a progenitor system have been considered and they were separated into two classes. In the single-degenerate scenario (SD-scenario) the

CO-WD accretes matter via Roch lobe overflow from a main sequence or red giant star (Whelan & Iben, 1973; Nomoto, 1982). During the accretion, the mass of the WD approaches the Chandrasekhar mass ($M_{\mathrm{Chan}} = 1.38\,M_{\odot}$; Chandrasekhar, 1931) limit, which is the theoretical stability limit of self gravitating system supported by the degeneracy pressure of electrons (see *e.g.,* Shapiro & Teukolsky, 1983). Subsequently, conditions in the center of the WD approach a state which enables thermonuclear burning of C-O. The actual explosion is preceded by a simmering phase (see Woosley & Weaver, 1986) in which almost all energy released by thermonuclear burning can still be efficiently transported away by convection but at some point this cooling mechanism becomes inefficient and the increase in temperature leads to a thermonuclear runaway and the onset of the explosion.

In contrast to that, the double-degenerate scenario (DD-scenario) involves two WDs in a binary system and their merger. In the classical interpretation this happens by slow accretion of the disrupted secondary onto the primary (Iben & Tutukov, 1984; Webbink, 1984). An explosion would then be triggered as the primary, again approaches the Chandrasekhar mass. However, it is believed that no thermonuclear runaway is induced by this process but rather an accretion induced collapse (Yoon et al., 2007). Recently the DD-Scenario has received interest again as it has been shown that a thermonuclear explosion can be triggered if the merger occurs violently, for a primary which is well below Chandrasekhar mass (Pakmor et al., 2010, 2012).

Another possibility to induce an explosion in a sub-Chandrasekhar mass model is a double detonation, where the primary WD accretes helium from its companion (Iben et al., 1987; Wang et al., 2013). This accretion does not proceed stably but at some point a detonation in the helium shell occurs, sending shocks into the C-O core which again create conditions suitable for the ignition of a thermonuclear explosion.

However, all scenarios have their problems and so far there is no consensus which ones are actually realized in nature. For example the SD-Scenario is in conflict with x-ray observations made (Gilfanov & Bogdán, 2010) and surviving companions should be observable but none have been found so far (Olling et al., 2015). For the DD-Scenario it is still a challenge for population synthesis to reproduce the observed SN rate (see Hillebrandt & Niemeyer, 2000). It is still not clear whether the double detonation scenario leads to SN Ia since the Helium ash distorts the synthetic spectra dramatically. With all these problems, of which this is only a small subset, it is still an open question whether the observations we make correspond to one scenario or a mix of all of them (*e.g.,* Woods & Gilfanov, 2013, 2014).

In addition to uncertainties about exact nature of the progenitor system, the details of how the thermonuclear burning process proceeds are also unknown. In principle, there are two ways in which a flame can propagate through the degenerate material, either by deflagration or by detonation. A deflagration proceeds subsonicly and energy transfers from hot ash to cold fuel via heat conduction. That means, the material in front of the burning front can react and expand. Thus, the burning happens in a less dense region and more intermediate mass elements (IMEs) are generated (see *e.g.,* Hillebrandt & Niemeyer,

2000). In the other mode, the detonation, the flame proceeds supersonicly, thus the cold fuel is heated through shock compression. Since the fuel cannot react to the supersonic flame front, burning proceeds at high densities and the fuel burns predominantly up to ^{56}Ni. In Chandrasekhar mass WDs, none of these burning modes in their pure form can lead to the explosions we observe. A pure deflagration explosion does not produce the ^{56}Ni masses we observe, while in a pure detonation model too much ^{56}Ni and not enough IMEs are created to match the observations. Thus, a combination of these two burning modes is investigated in a deflagration to detonation transition (DDT) (see Khokhlov, 1991) explosion in which nuclear burning is triggered in a deflagration flame, causing the WD to expand. At some point, the flame becomes supersonic and transitions into a detonation which rapidly disrupts the WD and still produce significant amounts of ^{56}Ni (Hillebrandt & Niemeyer, 2000).

1.2 Modeling Supernova explosions

1.2.1 Forward modeling

Theoretical attempts to reveal the nature of physical mechanisms that control the SN explosion can be separated in two classes, forward modeling and backward modeling. In the first approach, forward modeling, a theoretical model is followed through ignition and ejecta evolution by doing detailed hydrodynamic simulations with an accurate description of the burning processes (see *e.g.,* Reinecke et al., 2002). Then a link to observations is created by using detailed, multi-frequency, multidimensional radiative transfer codes (*e.g.,* Kromer & Sim, 2009) to compute synthetic observables. By confronting theoretical observations with real observational data, we can draw conclusions about the assumed model. The advantage of this method is a limited set of free parameters, however the simulation is time consuming and complex and thus not suitable for large parameter studies or the extensive analysis of different objects.

1.2.2 Backward modeling

In the case of the second approach, backward modeling, we typically start with observations and use much simpler, faster numerical techniques to produce synthetic observables Mazzali & Lucy (*e.g.,* 1993); Kerzendorf & Sim (*e.g.,* 2014). This is often done for entire grids of parameters in order to identify a set of parameters that matches the observations best (*e.g.,* Stehle et al., 2005). The downside of this approach is that many free parameters are involved and that the simulations often rely heavily on assumptions. Nevertheless, backward modeling can provide valuable and complementary information to forward modeling.

1.2.3 Abundance tomography

Specifically for SNe Ia, we can use the abundance tomography method introduced by Stehle et al. (2005) for backwards modeling. This method uses a spectral time series to reconstruct a spatially resolved composition of the SN ejecta. Here one relies on the fact that, as time progresses and the ejecta evolves, deeper layers of the ejecta become transparent and contribute to the formation of the spectrum. Thus, by successively fitting a model to spectra for different times, we can constrain the composition in these layers and thus perform a tomography of the SN ejecta. This method has been successfully applied to number of SNe, including SN 2002bo (see Stehle et al., 2005) and SN 2011fe (see Mazzali et al., 2015).

A big limitation of abundance tomography as it was done in the past is that the fitting is done manually and thus, it is possible to miss solutions. Additionally, degeneracies in the parameter space may pass unnoticed and the application of this method takes an excessive amount of time. With the abundance of computing power available today and the recent advancements in ML it is a great opportunity to apply these techniques to the abundance tomography method. In addition to the aspect of reducing the amount of human work that goes into applying this method, this opens up several new options. Firstly, automating the process allows us to analyze more data, gathered by surveys Li et al. (*e.g.,* 2000) which enables statistical interpretation of the results. However, more importantly by automating this approach, we can use algorithms that not only find the best composition but also give the topology of the parameter space and uncertainties. These informations are much needed to verify the correctness of this method.

In this thesis we present first steps towards the automation of this method. We present a framework that allows us to fully explore the parameter space of models in a timely manner while providing uncertainty estimates for the solution. Firstly, we introduce the techniques that are used for the framework in chapter 2. Then, in chapter 3, we discuss the tests performed on the framework and the results we obtained. Lastly, in chapter 4 we give a brief summary of our method and an outlook for the future.

2 Method

In this work, we present a novel way to perform abundance tomography, the approach of fitting an observed spectral time series to infer the composition, ejecta structure and density profile of an exploding star.

The original method, introduced by Stehle et al. (2005) to study SN 2002bo, relies on a fast radiative transfer method to predict synthetic spectra for simple SN models. The composition of the theoretical SN model is then changed and the theoretical spectra recalculated until a good fit to the observed spectral time series is obtained. Since this original method, which has been successfully applied to numerous other SNe (see Hachinger et al., 2009; Ashall et al., 2014; Mazzali et al., 2015) requires manual fine tuning of the model parameters (such as the abundances of the different chemical elements), an exploration of the entire parameter space is impossible and an assessment of potential degeneracies challenging.

We propose to automate this process with the help of ML techniques, such as an emulator (see section 2.3), a tool used to pre-calculate simulation results and interpolate between them and MCMC methods(see section 2.4.2). In addition to increasing the efficiency of the abundance tomography procedure, these tools allow us to incorporate prior knowledge about the nature of the exploding object such as nucleosynthesis constraints and provide detailed information of the solution topology.

This additional knowledge about our solution allows us to define regions in the parameter space which we can exclude. These intervals strongly depend on the measure of comparison, as we will discuss in detail in section 2.1. Because of that, the aim of this study is to find regions of parameter space we are confident to exclude and rule out models that lie within.

In this chapter, we will introduce all numerical techniques which are relevant for our automated approach to abundance tomography. We begin by discussing the details of fitting SNe spectra in section 2.1. Afterwards, we turn to TARDIS (Kerzendorf & Sim, 2014), the spectral synthesis code used throughout this work (section 2.2). Then we describe a way of accelerating the spectral synthesis process in section 2.3 and finish this chapter by outlining how the parameter space may be efficiently explored using MCMC methods in section 2.4.

2.1 Spectral fitting

It is impressive how astrophysicists are able to draw conclusions about the universe just based on observations of light. This is different to most other disciplines of physics which have access to experiments and laboratories to verify or falsify their theories. To compensate this, astrophysicists rely heavily on observations and try to catch the astrophysical systems in many different states and configurations. A crucial diagnostic in this aspect is the variation of the intensity of light with wavelength which is called, called the spectral energy distribution or short *spectrum*.

While observations are on the one side, theoretical predictions and the development of the models is on the other side. The models typically involve solving the coupled effects of many difficult physical processes. A task which is today most often performed on a computer.

2.1.1 Fitting SNe Ia

One field that has profited immensely from such a fitting process is the analysis of stellar spectra, including the realization that stellar spectra can be well described by only three parameters. These are the effective temperature, the surface gravity and the metallicity which describes the stars composition in relation to that of the sun. However, such a simple model does not work for SNe Ia. Their spectra do not feature a true thermal continuum on which lines are superimposed. Instead a multitude of lines from many species with no strong relation between them are blended together and form a pseudo-continuum. This already demonstrates, that a realistic model involves more than three parameters.

Already a one-dimensional description would at least involve of the order of ten parameters (see detailed description of our model in section 2.2.2), describing the abundances of the elements, the density and the radial velocity structure.

Unlike a star, a SN is a transient object that evolves on relatively short time scales. The time series of the spectra reveals additional information such that the parameters can in principle be constrained. Thus any procedure cannot rely on a precomputed grid such as in stellar spectroscopy (see *e.g.,* Husser et al., 2013) due to the high dimensionality of the parameter space. Thus, we have to use the simulation codes to generate a synthetic spectrum for the model parameters we want to study.

There are various methods available for simulating the processes in the SN ejecta. Simple tools like SYN++ (Thomas et al., 2011) are useful for line identification, but do not support solving the state of the plasma in the line forming region. Codes that include these more complex physics are for example Mazzali & Lucy (1993, ML93) code and TARDIS. They run a one-dimensional *Monte Carlo* (MC) simulation to iteratively solve the radiation field and the plasma state. The inclusion of additional physical processes come at the cost of a slightly longer run-time. However, they provide a good balance

between efficiency and accuracy, making them a good candidate for manual fitting of supernovae or abundance tomography. Therefore we will use TARDIS to generate the synthetic spectra needed throughout this thesis. The most accurate spectra can be generated with three-dimensional, time-dependent radiative transfer calculations. Tools like ARTIS (Kromer & Sim, 2009) or SEDONA (Kasen et al., 2006) take the output of a multidimensional hydrodynamic explosion simulation to create a full time series of the spectrum. One simulation takes roughly a week on a computing cluster, making them uninteresting for general fitting of supernovae. Their main application is the theoretical exploration of explosion models.

2.1.2 Comparing SN Ia spectra

In addition to a means to generate synthetic spectra, we need a procedure to compare them to observations.A key requirement here is a prescription to assess how well the synthetic spectrum matches the observation. For this measure we use a likelihood function which we will discuss in detail in section 2.1.3.

As a consequence, the radiative transfer calculations are never able to fully reflect the full physics and thus will be subject to systematic imperfections which may affect different parts of the spectrum differently. Therefore treating all features equally is often not advisable. Because of these issues, the comparison typically was so far done manually using the expert knowledge of trained astronomers who decided whether or not the model spectrum matches the observation. However, quantifying the measure requires to express the approach in an algorithmic way, which has not been done so far. We quickly outline the process of comparing spectra 'by eye' in the next paragraph.

chi-by-eye The challenge in comparing model spectra with observations lies in the imperfections introduced by assumptions and approximations in the model and the simulation code. As a result the important task is to decide which imperfections or discrepancies to accept. This requires extensive knowledge of the underlying physics and simulation code.

Applied to the task at hand, the general approach is to start by fitting the general shape of the SN spectrum and then account for the individual intensity of atomic lines. This works best for isolated and unsaturated lines, because the line depth at the core of the line is proportional to the amount of matter in the ejecta (see Hachinger, 2011, for a detailed explanation).

In addition to matching the general shape and individual lines, experienced astronomers also know what wavelength regions are prone to be systematically bad fits and can deweight these in their fitting approach. This allows an experienced astronomer to empirically identify regions where it is important to match the relative depth and profile of an atomic line instead of matching the continuum.

In summary, due to the large number of parameters (upwards of ten) and complex search space as well as non-formalized weighting of different spectral features, the process is very time consuming and subjective.

χ^2-measure A step beyond a visual inspection whether particular features match is to define a simple and quantifiable measure. The so called χ^2 measure is the most widespread approach::

$$\chi^2 = \sum_{i=1}^{N} \left(\frac{\theta_i - x_i}{\sigma_i} \right)^2 \tag{2.1}$$

as the distance to the expected result. Here $f(\theta_i)$ is the spectrum generated using model parameters θ_i (*e.g.,*Si abundance, O abundance, etc. for our problem), the observed spectrum is denoted as y_i and the uncertainty is σ_i. That means, for each bin, we determine the difference between the observation and our model scaled with the inverse of the uncertainty. We then square this and sum them all together. The result is a number that approaches the number of bins N for a perfect fit. If $\chi^2 << N$, this is called 'overfitting' and points to a flaw in the model or uncertainty because it is able to reproduce the data without the noise. On the other hand, $\chi^2 >> N$ is the result of an unsuccessful fit with a model that does not sufficiently reproduce the data (or an underestimation of the uncertainty in the data).

2.1.3 Likelihood

In the previous section we described ways to compare synthetic to observed spectra. We will now formalize this measure known as likelihood. This is the probability that the model θ results in the observations y we made which we will refer to as the posterior probability $P(\theta \mid y)$. We can use Bayes Theorem to express this in terms of the probability to observe y assuming model θ. This is called a likelihood function $P(y \mid \theta)$. To calculate the posterior probability

$$P(\theta \mid y) = \frac{P(y \mid \theta) P(\theta)}{P(y)}, \tag{2.2}$$

we additionally need an evidence $P(y)$ and a prior probability $P(\theta)$ (see *e.g.,* Jeffreys, 2011). In $P(\theta)$ we include the knowledge we have about the model and model parameters, *i.e.,*excluding combinations of parameters that are impossible to occur in nature. The term $P(y)$ is proportional to the observation y and thus equal for all models we compare. The important term in this formula is the likelihood function $P(y \mid \theta)$, which refers to the probability to make the observation y under the assumption of model θ.

As likelihood functions incorporate the model they are affected by its systematic uncertainty. While this often introduces difficulties in judging well-fitting solutions, in many cases it is still beneficial to rule out solutions that do not conform to the observations.

Because of that, our goal is to quantify the area of parameter space we can exclude with a high confidence, instead of giving an interval of values as the 'solution'.

The big advantage of using Bayes Theorem is that we can include prior probability in the analysis of the problem. We are not using the full potential of this approach because we use a uniform prior of zero or one (for details see section 2.2.2).

Because it simplifies computation, we often use the natural logarithm of all probability distributions: the log-likelihood $L(y \mid \theta)$, the log-posterior $L(\theta \mid y)$ and the log-prior $L(\theta)$.

χ^2 - Likelihood In section 2.1.3 we define the χ^2 measure, which is the main component of the popular least-squares fitting approach. The common χ^2 is simply the natural logarithm of the likelihood:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right), \tag{2.3}$$

with variance σ and mean μ, we have defined the probability distribution of x. We apply this to a spectrum with N bins, an uncertainty σ_i with $i = 1, \ldots, N$ and a mean μ_i in each bin. The resulting probability is the product of the probabilities in each bin. We use the natural logarithm of eq. (2.3) to transform this product into a sum:

$$L(x \mid \theta) = -\frac{1}{2}\sum_{i=1}^{N}\left[\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2 + \ln(2\pi\sigma_i^2)\right]. \tag{2.4}$$

If we define x as our model spectrum $f_i(\theta)$ and μ_i as the observation y_i, the first part of the summation in eq. (2.4) reads the same as the χ^2 measure defined in section 2.1.2. The only change compared to eq. (2.1) is the addition of the normalization term.

However, applied to SN Ia spectra this likelihood does not perform well as systematic errors introduced by the simulation code cause the synthetic spectrum to differ from the observation by several σ. For the statistical error we use a rough estimate of the photon noise and use an estimate of about 5% of the observation. To solve the problem of the systematic error we use a trick by Hogg et al. (2010) to estimate the systematic error during the exploration of the parameter space. As a result, we define

$$\sigma_i^2 = \left(\frac{5}{100}\bar{y}\right)^2 + f_i(\theta)^2 \cdot e^{2\gamma} \tag{2.5}$$

as our uncertainty with $\bar{y}$ as the mean of the spectrum and γ as the parameter that controls the systematic error. We introduce γ is a nuisance parameter, which we will marginalize over during the analysis.

Other likelihoods In search for a likelihood, that works well for SN Ia spectra, we explore several approaches to modify the spectra prior to calculating the likelihood, , *i.e.,*using χ^2 on the derivative of the spectrum or subtracting a polynom. We discuss them in detail in section 3.3.

For stellar spectra Czekala et al. (2015) defined a flexible likelihood using a combination of local and global Gaussian Process (GP) kernels to estimate the covariance and therefore account for uncertainties in the model (see section 2.3.3 for details on GPs). Using a similar approach for SN spectra is beyond the scope of this work and involves the quantification of the systematic errors in the SNe models. Additionally, the suggested approach takes a significant amount of computation time because the process of fitting the GP kernels has to be repeated for every evaluation of the likelihood. With a reported runtime of 2 h, this approach is not feasible for parameter space exploration.

An interesting approach to this problem is a metric space for SNe Ia, an idea pursued by Sasdelli et al. (2014). This space, constructed from observations using a Principal Component Analysis (PCA) (for details see section 2.3.2) of the spectra to reduce the dimensionality and associate features with dimensions, would enable a likelihood directly from the principal components. This however requires that the synthetic spectra are sufficiently close to the observations which went into construction the PCA space.

The staring point for a likelihood tailored to comparing SN Ia would be a χ^2 modified with ideas by Sasdelli et al. (2014) and Czekala et al. (2015). Instead of using the spectrum for comparison, one can enhance line features by taking the derivative or by subtracting the continuum. We do a quick test of several ideas in section 3.3. Exploring these ideas further is beyond the scope of this thesis.

2.2 Tardis

All synthetic spectra used throughout this thesis are calculated with TARDIS (Kerzendorf & Sim, 2014), a modular and highly configurable spectral synthesis code for SNe utilizing MC techniques. The code is mostly written in PYTHON with the computationally heavy parts implemented in C.

To create a spectrum from a model, we have to solve the radiative transfer equations. A fast way to do this, is to use MC quanta to discretize the radiation field. These MC packets represent units of indivisible radiative energy, a formalism proposed by Abbott & Lucy (1985); Lucy (1999) to ensure radiative equilibrium throughout the simulation. The packets propagate through the SN ejecta and randomly encounter bound-bound or bound-free interactions, changing its direction and co-moving frame frequency.

Because the ejecta is gravitational unbound, we can assume free, homologous expansion. The simulation is not time dependent and limited to one dimension. The computational domain is spherical symmetric and discretizes the radial coordinate into shells. Packets traveling outwards and leaving the outermost shell are recorded to form the spectrum. Following Mazzali & Lucy (1993); Lucy (1999), we assume a Schuster-Schwarzschild boundary on the inside. That means all energy generation, for example through radioactivity, happens below this boundary (we will refer to this in the following as *the photosphere*) which emits radiation with a thermal spectral energy distribution. This assumption is good enough for normal SN Ia, even though they have no thermal continuum. Because the thermalization layer is much deeper inside than the photosphere a pseudo-continuum is formed. Nevertheless it has been shown that this crude simplification yields results that are in good agreement with observations (Mazzali & Lucy, 1993; Lucy, 1999; Mazzali, 2000) As a consequence, we initialize the MC quanta at the inner boundary with a frequency distribution according to Planck's law.

To provide the probabilities needed for the MC simulation, TARDIS solves the plasma state. For this purpose, TARDIS assumes that the main influence on both ionization and excitation comes from the radiation field. With the help of two estimators collected during the MC simulation, it is possible to update the ionization fractions, the level populations and the radiative rates. For the calculation of the ionization fractions TARDIS uses a nebular approximation (see Eq. (3) in Kerzendorf & Sim, 2014) proposed by Mazzali & Lucy (1993). The level populations are calculated using the DILUTE-LTE approach (see Eq. (4) and (5) in Kerzendorf & Sim, 2014) employing a basic non-local thermodynamical equilibrium (NLTE) approximation for excited states and local thermodynamic equilibrium (LTE) otherwise. To obtain the radiative rates, TARDIS uses the simple radiation field model to link the mean intensity at the blue wing of the transition between levels l and u

$$J^b_{lu} = W B_{\nu_{lu}}(T_R). \tag{2.6}$$

to the Planck function $B_{\nu_{lu}}$ and parameters of the radiation field: the radiation temperature (T_R) and a *dilution factor* (W), which relates the radiation field to the theoretical

blackbody (see Mihalas, 1978). These are iteratively updated from the estimators of the MC simulation in each grid cell. The radiative rates are then calculated using Eq. (6) and (7) in Kerzendorf & Sim (2014).

During the propagation the packet can undergo two types of interactions - bound-free Thomson scattering by free electrons and bound-bound interactions with atoms and ions. Thomson scattering is considered as a coherent scattering process, thus only the propagation direction of the packet changes, not its co-moving frame frequency.

The main source for opacity in the SN ejecta are bound-bound interactions which are treated using the Sobolev approximation (see *e.g.,* Lamers & Cassinelli, 2015). We use a macro atom scheme to simulate the re-emission of an absorbed packet. The bound-bound transition activates a macro atom to the upper level of the absorbing transition. After undergoing a series of internal transitions, the macro atom deactivates and re-emits the energy in form of a new MC packet to the simulation.

In general a TARDIS calculation consists of two parts. Since the plasma state depends on the radiation field itself, a number of simulation cycles (typically 20) are performed first to calculate a self-consistent plasma state iteratively. This plasma state is then used in a final high resolution cycle to calculate the emergent spectrum. The full simulation for one model takes an average of 100 s on a machine with eight cores and outliers take up to twice that time. The execution time heavily depends on the abundance of elements with thousands of possible line transitions, most notably iron.

2.2.1 Explosion model

An explosion model defines the structure and composition of the exploding star. Typically these models come from hydrodynamic simulations and provide the density profile abundances needed for the radiative transfer simulations. We base our simulations on the W7 model Nomoto et al. (1984) which is a one-dimensional deflagration model where the rate of burning was modified to explain the observables of normal SN Ia well. We use a power law approximation proposed by Branch et al. (1985) to obtain the density profile for our calculations (see fig. 2.1). We discretize the model into 20 regions, based on the radial velocity and take the mean density of each region. However, we do not copy the composition of the W7 model but use a uniform abundance structure as our key parameters which are described in the next section.

2.2.2 Model Parameters

We use a relatively simple model in this work containing only 13 parameters. – Eleven of those control the uniform abundance structure of the ejecta (see table 2.1). They are

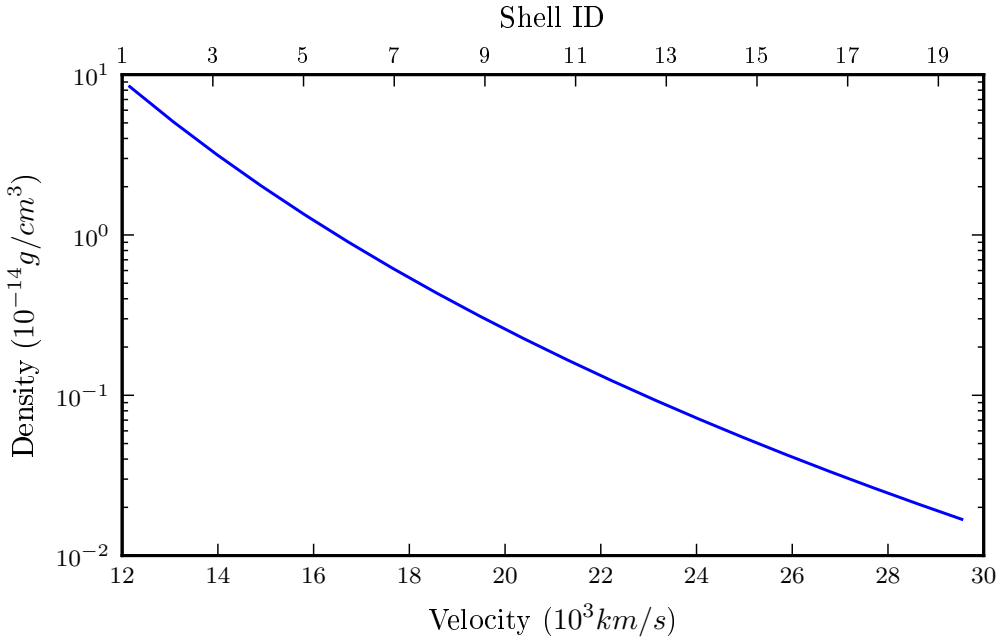

Figure 2.1: Power law approximation of the density profile as proposed by Branch et al. (1985) for the W7 model (see Nomoto et al., 1984). This is used throughout this work as the input density profile for all our calculations.

given as atomic mass ratios X_n, where n is the element. These are absolute ratios, *i.e.*,all X_n have to sum up to one:

$$\sum_{n=1}^{N} X_n = 1. \tag{2.7}$$

To always fulfill this requirement, we calculate one ratio based on all others. We choose oxygen as this filler ratio because its impact on the spectrum is rather low. A positive oxygen abundance is our first prior requirement for the parameters. We will discuss priors in detail in the next section.

The other two parameters are the temperature and velocity of the photosphere. The temperature of the photosphere T_i is at the same time the temperature of the blackbody emitted by the inner boundary. The last model parameter is the velocity of the photosphere v_i, which accounts for the position of the photosphere in the ejecta. Whenever we mention model parameters, we will denote them as $\vec{\theta}$. Depending on the context, this may refer to a subset of all parameters.

For all parameters introduced in this section we can define priors (see section 2.1.3). We require the prior to have zero probability if any of the abundances have values outside 0 to 1. Due to nucleosynthetic predictions we also define our prior to have zero probability

Table 2.1: Upper limits for the abundance parameters from observations and based on nucleosynthesis calculations.

Element n	Si	S	Ca	Fe	Co	Ni	Mg	Ti	Cr	C	O
$X_{n,max}$	1.0	0.5	0.1	0.2	0.2	0.2	0.1	0.02	0.02	0.1	1.0

if the ratio of sulfur abundance is greater than the silicon abundance. For all other elements we define an upper boundary (*e.g.*,0.5 for X_{S}) for the abundance.

As noted earlier, we require a valid oxygen ratio as described in the previous section. Further, we can use nucleosynthetic predictions to exclude models with a higher sulfur than silicon ratio. For all other elements we define an upper boundary based on empirical data from observations and hydrodynamic simulations. These are shown in table 2.1.

For models with T_{i} and v_{i} as parameters, there is another prior that ensures that the model is theoretically able to emit the luminosity we observe. For this we require

$$0.9\,L < 4\pi\sigma_{SB}\,(v_{\mathrm{ph}}t)^2\,T_{\mathrm{ph}}^4 \tag{2.8}$$

with L as the total luminosity of the observed spectrum, σ_{SB} the Stefan-Boltzmann constant, v_{ph} the velocity of the photosphere, t the time since explosion and T_{ph} the temperature of the photosphere.

2.3 Spectral emulator

For our goal, the reconstruction of the explosion of a supernova, it is necessary to compare many thousands to millions of evaluations of a model to find sets of parameters that reproduce spectra similar to the observation. We try to restrict the parameter space we cover during our search as little as possible, by using broad boundaries (see priors in section 2.2.2). However, even with the fastest simulation codes available, making heavy use of approximations to increase performance, the whole analysis becomes very time-consuming. The analysis of a single observed spectrum would take roughly one month on a computing cluster.The limiting factor is the speed at which we can generate spectra.

However, we can make the generation of spectra much more efficient if we create a grid of synthetic spectra and use interpolation techniques to approximate values between precomputed points (*e.g.*,linear interpolation with STARKIT (Do et al., 2015)). The significant increase in speed for spectra generation allows us to continue the analysis and explore the parameter space. However in multiple dimensions, linear interpolation is not ideal because a prohibitively large number of points would be needed for the gradients to be sufficiently enough resolved for this method to yield acceptable results.

Naturally interpolation is associated with uncertainties because we delve into areas where no actual calculations were performed. For stellar spectra, interpolation was explored by Czekala et al. (2015) with the goal to quantify these uncertainties. They created a tool,

which processes the precomputed synthetic spectra in a particular way to allow for the interpolation of stellar spectra. Combining these two techniques as so called emulator framework was created that acts like an actual simulation code.

In this section we describe how we developed a similar process for the use with SN Ia spectra. We aim to create a spectral 'emulator', that uses a precomputed grid to mimic the behavior of TARDIS for the creation of SN spectra. An overview of the processes involved is shown in fig. 2.2.

The emulator has three main parts with different contributions to its performance:

Grid The models we try to fit, have many parameters (from ten up to 50) which translates into a high-dimensional parameter space. At the same time, the relative part of the parameter space that is interesting to the analysis shrinks. A sparse grid could easily miss this area but a uniform grid that is dense enough would require too much resources to create. Thus we need an efficient way to generate a grid that covers the whole range for all parameters. One method is a Latin Hypercube (Stein, 1987) (LH) that places grid points in a non-uniform way such that all values for one parameter are used once and that the distance between points is maximized. The details are outlined in section 2.3.1.

Preparation Because the synthetic spectra have a poor S/N ratio, we use a smoothing algorithm (Savitzky-Golay (Savitzky & Golay, 1964) filter (SG filter)). To construct the interpolation basis, we create a set of eigenspectra with the help of a Principal Component Analysis (PCA). This is done to reduce the number of dimensions to interpolate and to have smooth changes in the values to interpolate (see section 2.3.2 for details).

Interpolation Following the approach by Czekala et al. (2015) we use a Gaussian Process (GP) to interpolate in the basis of the eigenspectra obtained by the PCA where the values vary reasonably smooth. The interpolation for one set of model parameters then returns weights which are used in a linear combination of the eigenspectra to create a spectrum.

2.3.1 Grid

The interpolation requires a grid of spectra to interpolate between. For the creation of the spectra we run TARDIS for each grid point. However, a uniform grid is not an option because of the required number of dimensions, the grid would either be too sparse or creating the spectra for it would take unreasonable amounts of time. Thus, we need a non-uniform grid, that covers the full spectrum of values for the parameters while at the same time spreading all values equally across multiple dimensions. One algorithm that fulfills these requirements is a Latin Hypercube (Stein, 1987) (LH). This algorithm only needs the boundaries of all dimensions to create a grid with N points. Each dimension is split into N values and points are chosen by drawing a random value for each dimension

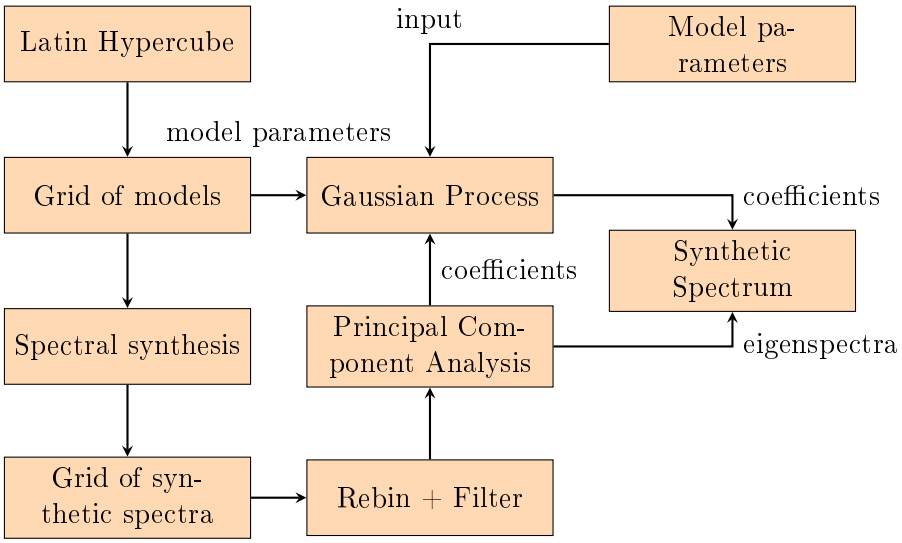

Figure 2.2: Illustration of the main processes building up our SNe Ia spectral emulator.

without picking the same value twice. The result is a grid of N points that guarantees all values are chosen once for each parameter. We repeat this process 1000 times and select the grid with the highest distance between the two closest points is used. This way we select a grid that is spread equally. We use this approach to generate a grid with values ranging from 0 to 1 and later transform these into model parameters.

2.3.2 Preprocessing

In principle interpolation between the raw spectra is possible. However, this would require a dense sampling. As described previously the high dimensionality of the parameter space allows only a sparse sampling where interpolation on the raw spectra is not possible. Interpolation requires a smooth variation of reference values. The interpolation process is additionally complicated by the MC noise in the reference values which implicated that these spectra do not vary smoothly as a function of the input parameters. In this section we discuss the steps we take to transform the spectra at the grid points, our spectral grid, into a form that allows us to easily interpolate them.

First, we change the binning of all spectra so the bins are uniform in $\log \lambda$. Because of the first order Doppler formula, a uniform velocity interval

$$\frac{\Delta v}{c} = \frac{\Delta \lambda}{\lambda} = \Delta \log \lambda \tag{2.9}$$

is associated with a uniform interval in $\log \lambda$. Thus using $\log \lambda$ intervals we have a uniform resolution in 'velocity space'. Line profiles are determined by the projected velocity, therefore the shapes of line profiles are equally well resolved, regardless of which part of the spectrum they fall in (priv. comm. S. Sim).

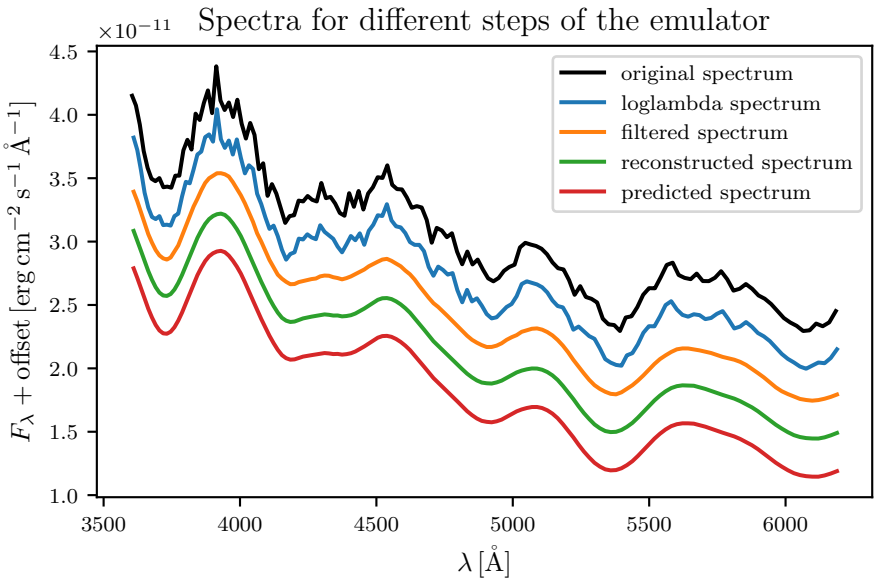

Figure 2.3: Plot showing various stages of preprocessing: rebinning (loglambda spectrum), SG filter (filtered spectrum), PCA reconstruction (reconstructed spectrum) and emulator prediction (predicted spectrum).

Savitzky-Golay filter The synthetic spectra generated by TARDIS contain MC noise. However, the PCA works best when the spectra share many common features, such as atomic lines. Noise is a problem in this case, as it interferes with the detection of features. Subsequently, the PCA needs more eigenspectra to represent the dataset, most of which are dominated by noise.

To smooth the spectra, we use a SG filter, a popular digital filter to process data with low S/N. The filter does a least squares fit with a low order polynomial to data-points adjacent to the point of interest and evaluates it at the center. The filter has two parameters that need to be optimized by hand. They mainly depend on the shape and S/N of the spectra grid. The order of the polynomial controls the degree of smoothing: The lower the order, the smoother is the output. The second parameter is the window length, that is the number of adjacent points, that are included in the least squares fit. Increasing the window length puts more constraints on the polynomial, making it better behaved and increasing the smoothness. In section 3.2.1 we look at different parameters for the filter and determine the values we use throughout this thesis.

Principal Component Analysis Alongside the Gaussian Process (GP) (see section 2.3.3) the Principal Component Analysis (PCA) (see *e.g.,* Jolliffe, 2002) is one core component of the emulator approach. As we will discuss in section 2.3.3 in detail, for the interpolation to work best, the values to interpolate should vary smoothly. In our case, due to the

sparseness of the grid, the spectra which are mainly dominated by line interactions do not fulfill this requirement. For our approach the PCA fulfills two purposes: We use it to reduce the dimensionality so the GP is able to interpolate the space. Secondly, doing the transformation yields a space where the values vary much smoother.

We follow Czekala et al. (2015) and standardize the spectral grid $\vec{F}$ by subtracting the mean spectrum ξ_μ. We use these standardize spectra to create a full set of eigenvectors ξ that we combine into a $N_{\text{bin}} \times N_{\text{bin}}$ matrix $\vec{\xi}$ where N_{bin} is the number of bins in the spectral grid. We calculate the modified spectral grid

$$\vec{\hat{F}}_k = \vec{F} - \xi_\mu - \sum_{i=1}^{k-1} \vec{F} \xi_k \xi_k^T, \tag{2.10}$$

that we use for determining the k-th eigenspectrum ξ_k. In other words, for the calulation of the k-th component we subtract the reconstruction of all previous components from all spectra in the grid. We then maximize

$$\frac{\xi^T \vec{F}^T \vec{F} \xi}{\xi^T \xi} \tag{2.11}$$

with $\vec{F} = \vec{\hat{F}}_k$ to obtain a new eigenvector $\xi_k = \xi$. We repeat eq. (2.10) and eq. (2.11) for $k = 1, \ldots, N_{\text{bin}}$ to calculate the eigenspectra matrix $\vec{\xi}$. With this matrix we can transform all spectra in the grid into the eigenvector basis $\vec{\xi}$ to obtain the coefficients

$$\vec{C} = \left(\vec{F} - \xi_\mu\right) \vec{\xi}. \tag{2.12}$$

We can use these coefficients in a linear combination of the eigenspectra to fully reconstruct all spectra

$$\vec{F} = \xi_\mu + \vec{C} \vec{\xi}^T. \tag{2.13}$$

with $\vec{C}$ as the matrix of coefficients for all spectra in the grid.

Due to the way the PCA calculates the eigenspectra, the elements of $\vec{\xi}$ are in decreasing order of 'importance'. In this context importance means that higher order components encode less variation of input data. Thus, truncating the series at the right element allows us to reduce the dimension without the loss of information. Specifically, we truncate after N_{comp} elements, a value we determine manually prior to the analysis based on the spectral grid. $\vec{\xi}_{N_{\text{comp}}}$ denotes the $N_{\text{bin}} \times N_{\text{comp}}$ matrix that forms the basis for the space we will later interpolate in. We will interpolate the truncated components

$$\vec{C}_{N_{\text{comp}}} = \left(\vec{F} - \xi_\mu\right) \vec{\xi}_{N_{\text{comp}}}. \tag{2.14}$$

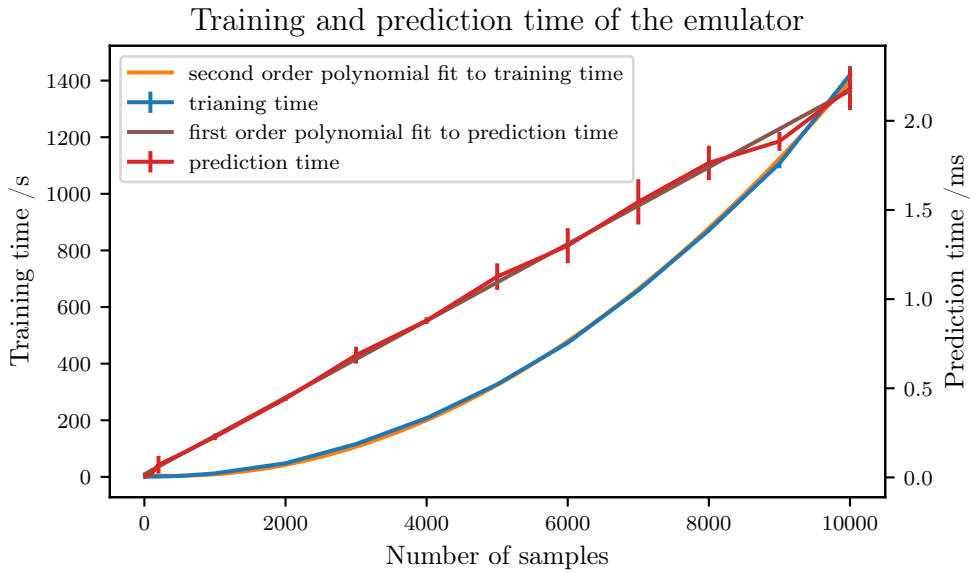

Figure 2.4: Illustration of the physical time it takes to train and emulate the emulator as a function of the size of the input data. Note that the prediction time is given in milliseconds on the right y-axis and the training time is given in seconds on the left y-axis.

For clarity we will drop the subscript N_{comp} in the future and assume all values in PCA-space to be truncated to N_{comp}, *i.e.*,$\vec{C} = \vec{C}_{N_{\text{comp}}}$.

As described earlier, the PCA has one parameter: the number of components N_{comp}, which we optimize manually. There are automated ways of determining this value by requiring that the difference between the original and the reconstruction falls below a threshold for all spectra. However, our tests showed that determining a reasonable threshold is equally difficult to determining N_{comp} directly, so we did that. Optimizing the number of components is again a balance between speed (due to more values to interpolate) and accuracy (due to the loss of information when truncating).

The PCA proves to be a useful tool that solves two problems at once. While producing good results, this method has an inherent drawback because the features defined by the PCA are not a function of the model parameters. As we will discuss in section 3.2.3, there are model parameters that result in a shift of the features instead of a change in amplitude. Such features are not possible to describe with a PCA.

2.3.3 Interpolation

As detailed in section 2.1, comparing synthetic spectra with observations is a powerful tool to measure quantities like effective temperature or metallicity. However, the generation of spectra for certain parameter sets often requires an expensive simulation. Thus to quickly obtain spectra for arbitrary parameter sets one uses interpolation between precomputed grids.

As detailed in above, we need more complex techniques than linear interpolation, since we are dealing with large datasets with a non-uniform distribution of points. Czekala et al. (2015) suggest a two step process – not interpolating on each spectral points but on the coefficients for eigenspectra (see section 2.3.2) and using an interpolation technique that allows us to estimate the uncertainty in the interpolation (*e.g.,*giving a larger uncertainty for a point that is more distant from grid points than for a point that is densely surrounded by grid points).

Specifically, we follow Czekala et al. (2015) and first perform a PCA, then determine how many eigenspectra we need and then use a so called GP (see section 2.3.3) to interpolate as well as get uncertainties between them. However propagating these uncertainties through the PCA is difficult. Czekala et al. (2015) instead draw Gaussian random variables for the coefficients and used these to reconstruct spectra. These reconstructed spectra represent the variance of all possible spectra for this set of model parameters.

Gaussian process A Gaussian Process (GP) is a statistical model that assumes observations in a continuous space are normally distributed random variables Rasmussen & Williams (2006). A collection of those random variables has a multivariate normal distribution with a covariance depending on the similarity between two points θ and θ'. The similarity is defined by a kernel function. The covariance between all observed points forms the matrix $K(\theta, \theta')$. If this depends on parameters, so called hyperparameters, these need to be optimized before the GP can be used for interpolation. We use a squared exponential kernel to describe the similarity with

$$K(\phi, \theta, \theta') = a^2 \prod_{n=1}^{N} \exp\left[-\frac{(\theta_n - \theta'_n)^2}{2\, l_n^2}\right], \tag{2.15}$$

where N is the number of input parameters, *i.e.,*the dimension of θ and $\phi = \{a, l_1, \ldots, l_N\}$ is the collection of hyperparameters. Specifically a defines the amplitude while l_n define a length-scale for the n-th input dimension. The length-scale is the extend to which points influence each other. Optimizing ϕ defines the GP completely and allows prediction, *i.e.,*interpolating and estimating an uncertainty.

Using a GP for data prediction is a Gaussian process regression problem that is solved by the Kriging equations (see *e.g.,* Cressie, 2015). They define the mean estimate as

$$\mu = K(\phi, \theta^*, \theta) K(\phi, \theta, \theta')^{-1} f(\theta) \tag{2.16}$$

and the variance of the estimate as

$$\mathrm{Var} = K(\phi, \theta^*, \theta^*) - K(\phi, \theta^*, \theta) K(\phi, \theta, \theta')^{-1} K(\phi, \theta^*, \theta)^T, \tag{2.17}$$

where $K(\phi, \theta^*, \theta')$ is the covariance between the new point of estimation $\theta*$ and all other observed points θ for a given hyperparameter vector ϕ, $K(\phi, \theta, \theta')$ is the covariance between all observed points, $f(\theta)$ is the value of the observed sample at point θ and $K(\phi, \theta^*, \theta^*)$ is the variance at point θ^* as dictated by ϕ. A notable drawback of GPs is $\mathcal{O}(N^2)$ scaling of the prediction time with the number of observed points. In fig. 2.4 we show a comparison of how long the training process takes and how long one prediction takes. The $\mathcal{O}(N^2)$ scaling is clearly visible: the 2nd order polynomial fits perfectly. For the evaluation time we see a $\mathcal{O}(N)$ scaling instead of $\mathcal{O}(N^2)$ because we are not yet utilizing the uncertainty due to reasons we outline below. Overall the scaling makes working with large datasets slow and fuels the search for alternatives.

Summarizing, we can interpret the probability distribution for the coefficients as a N_{comp}-dimensional Gaussian distribution whose mean and covariance are functions of the point θ^*, the hyperparameters ϕ and the (fixed) values of the coefficients at grid points (see Czekala et al., 2015).

Application Applying the method of GPs to our approach, we already established to interpolate the coefficients of the spectra in the basis created by the eigenspectra of the PCA. That means the observed values $f(\theta)$ are the coefficients C (see section 2.3.2) and the grid points are the θ that are used by the GP.

Although we generally followed the approach by Czekala et al. (2015) in constructing the emulator we made adjustments to the detailed workings of the emulator.

The first notable change is that we use one GP to model all coefficients, while they advocate the use of one GP per coefficient. A comparison between an unoptimized implementation of the latter to the former showed no significant improvement in the quality of the prediction but came at the cost of a N_{comp}-fold increase in execution time for the training phase as well as the prediction. For the tested case, the training took over a week, compared to approximately 2 h for the version with a single GP. Pursuing this idea further is out of scope for this thesis as the chosen approach works well enough.

Whilst being one of the main reasons we chose GP interpolation, the current implementation does not yet make use of the predicted uncertainties. The reasons being a notable increase in evaluation time for the prediction and that this does not work well with only one GP.

These are the main points that differentiate our approach from what Czekala et al. (2015) described. For the fitting of the GP we provide the grid of points and the eigenspectra

coefficients. We also provide sensible boundaries for ϕ to keep the values in the range we expect, *i.e.*,$l_n \in [10^{-3}, 10^2]$. This prevents, for example, length scales below the average distance between grid points, which would in the interpolation only work in the direct vicinity of grid points.

Summarizing, our approach works quite well and we made some adjustments to reduce the runtime to fit our needs. The inherently bad scaling of the GP in combination with the requirement of a finely sampled grid and the wish to limit the search space as little as possible has created a setting where GP interpolation meets its limits. Future work might push these limits back with advanced techniques for the selection of the grid used to fit the GP. We will discuss these options in section 4.2.

We conclude this section with an important technical detail. Czekala et al. (2015) describe the usage of a 'nugget' term that accounts for the imperfections in the spectrum introduced by the PCA analysis. In particular, this term is needed to allow for coefficients at grid points to be outside the range predicted by the GP which may happen due to noise or abrupt changes in the coefficients. Without this parameter, the optimization of the hyperparameters ϕ can easily fail if coefficients (see eq. (2.14)) are in conflict to another. While they optimize this term in conjunction with the other hyperparameters, we use a constant 'nugget' of 10^{-4}.

2.4 Exploring the parameter space

One way to explore the parameter space is to do it by hand. We already gave a quick overview of this procedure in section 1.2.3. Although this is a slow process, this approach yields good results (see *e.g.,* Stehle et al., 2005; Hachinger et al., 2009; Mazzali et al., 2015). However, quantifying the results and revealing correlations between parameters cannot be done manually. Additionally it is much easier for a human to miss important areas than for an algorithm that proceeds systematically.

The first step towards automation is to use an optimizer to find the best fitting set of parameters. However, a study by Jancauskas et al, in prep. showed that repeating this process with different initial conditions yields nearly identical spectra but with some parameters differing by a factor of two to three (*e.g.,*the cobalt abundance). This degeneracy in the parameter space creates doubt in the viability of this approach. Past analysis' might be affected by this phenomenon without being noticed. Thus, it is important to visualize the distribution of models and quantify the spread of the parameters. We use a posterior function to quantify the similarity between the observation and a model spectrum. The distribution of this posterior function, the posterior probability density function (PDF), can give us insights about the parameter space.

We can visualize the PDF and thereby the parameter space, by approximating the PDF and drawing points from its distribution. The PDF represents how probable one set of model parameters is to match the observation based on the posterior probability. A

popular and efficient method to approximate the posterior PDF is Markov chain Monte Carlo (MCMC). The details of this process are outlined in section 2.4.2.

2.4.1 Optimization

In astrophysics it is often required to numerically optimize parameters. That is to find the best values that minimize or maximize a function. For example, a simple optimization problem is fitting effective temperature to the spectrum of a star by minimizing the χ^2 of the synthetic spectrum and the observation and thus finding the maximum likelihood for a spectrum.

In general we distinguish between two classes of optimization algorithms. Local optimization is a fast way to find optimal values, starting from a guess. An example method is the popular Levenberg-Marquardt algorithm (see *e.g.,* Nocedal & Wright, 2006). The disadvantage of these methods is that they stop once they found an optimal point at which a step in any direction would result in a decrease of the function's value. That means, there is no guarantee to find the globally best value. For applications with only a single peak, this is fine, however many applications involve noise, spikes in the data or periodicity where local optimization fails.

Global optimization aims to find the best value in a specified range. At the cost of speed, these methods aim to probe the parameter space more thoroughly and find the globally best value. This is especially useful if there are several peaks in the parameter space. One algorithm that implements global optimization is Differential Evolution (differential evolution) (see *e.g.,* Chakraborty, 2008). For examle Jancauskas et al, in prep. found multiple solutions using differential evolution, a global optimization algorithm, with a similar χ^2 and spectra, but different model parameters (see section above). This result suggests strong degeneracies in the parameter space.

2.4.2 Markov-Chain Monte Carlo

While optimization algorithms are able to, very efficiently, find the optimum value of a function, they generally fail to provide additional, much needed information. For example, if there are multiple solutions that are almost equally good for the observations at hand. In section 2.1.3 we define a posterior probability that assigns to every point of the parameter space a probability of how likely it is for a model with these parameters to produce the observations. This is referred to as the probability density function (PDF).

In the previous section we discussed, how the PDF can be used to find the best combination of parameters. However, degeneracies in the parameter space require us to look at the region surrounding the optimum. The challenge in exploring the parameter space is to obtain a good approximation to the PDF, which allows us to visualize the topology of the parameter space.

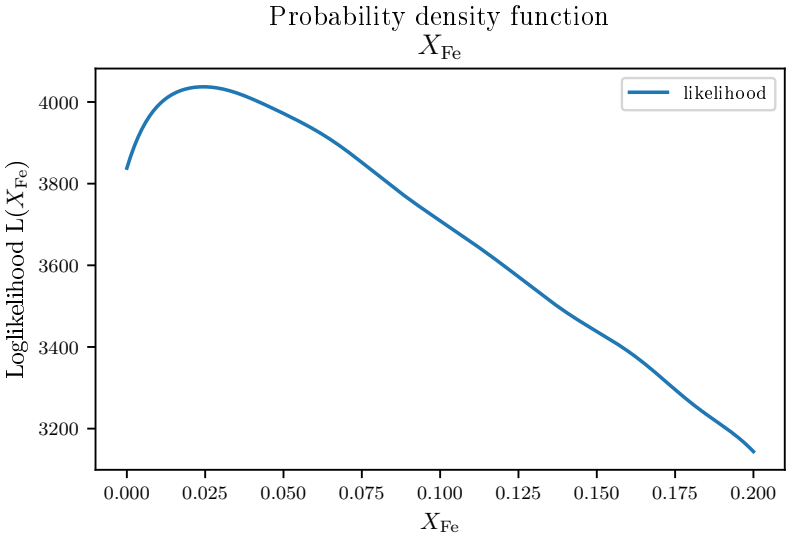

Figure 2.5: Plot showing the PDF associated with X_{Fe}. Showcased with the dataset from section 3.2.1 and a basic χ^2 (see eq. (3.5)).

There is an efficient method to approximate a PDF, namely the Markov chain Monte Carlo (MCMC) technique. The general idea of this approach is to start from a proposal PDF that is successfully refined to converge towards the target PDF. This is done by drawing samples from the PDF and use Bayesian Inference (see *e.g.,* Sivia & Skilling, 2006) and their posterior probability of the samples to iteratively update the approximation. After an initialization phase (*burn-in*), convergence is achieved and drawing from the approximation yields samples that are compatible with the posterior PDF. Thus, after the burn-in the parameter space can be efficiently explored by continuously drawing sampled from the PDF.

Different implementations of MCMC differ mainly in their proposal function, that is how the algorithm chooses the next state based on the current state. A famous example is the Metropol-Hastings algorithm (see Hastings, 1970), however in this work we use the EMCEE (see Foreman-Mackey et al. (2013) (EMCEE) implementation of MCMC because it has proven to work well in many dimensions in addition to being efficient in the generation of uncorrelated samples.

To advance the state in a MCMC application, a new candidate state is found by evaluating the proposal function. Then the posterior probability of the candidate position P_{new} and the current position P_{old} is calculated and compared. The candidate is always accepted, *i.e.,*the candidate becomes the current point, if

$$P_{\mathrm{new}} > P_{\mathrm{old}} \tag{2.18}$$

otherwise we only accept the candidate with the probability

$$P_{\text{accept}} = \frac{P_{\text{new}}}{P_{\text{old}}}. \tag{2.19}$$

The states used by MCMC retain in general information from one step to the next resulting in correlations in the history of points. We can calculate an autocorrelation time, that is the number of steps it takes for a chain to have an independent state from an arbitrary starting point (see Sokal, 1996, p. 16 for details). For the purpose of exploring the parameter space only the number of independent samples is relevant, thus we skip intermediate ones.

Markov-chain Monte Carlo Ensemble Sampler In this section we give a brief overview of the implementation of the MCMC algorithm using the ensemble sampler EMCEE developed by Foreman-Mackey et al. (2013) and proposed by Goodman & Weare (2010). The state of this implementation consists of an ensemble of N positions that each form an individual Markov chain. New candidates are proposed based on the current positions of these chains (*walkers*).

All walkers are propagated simultaneously. For this purpose, the ensemble is split into two equal subsets and each chain is randomly paired with one from the other subset (this pairing is not bi-directional). A new candidate position C is found based on the current positions of the pair A and B according to

$$C = B - x\,(B - A)\,. \tag{2.20}$$

where x is determined in the random experiment

$$x = \frac{(u+1)^2}{2}, \tag{2.21}$$

using a random number u, uniformly distributed over [0, 1). To compare the likelihood at the candidate and the current position, we modify P_{new} such that

$$\hat{P}_{\text{new}} = x^{d-1} P_{\text{new}} \tag{2.22}$$

includes a term reflecting the influence of x, where d is the number of dimensions.

If we apply this method to our approach, we generally start with random positions following our prior distribution. We then iterate the chains long enough for the chains to converge on the important area. This takes approximately 5×10^3 iterations. Afterwards, we use the converged state of the sampler to draw the points we will use to visualize the parameter space. For this we run between 2×10^4 and 10^5 iterations. Based on these samples we calculate an autocorrelation time and use it to filter out correlated samples. Thus the remaining, independent samples correctly reflect the posterior distribution.

3 Results

It is our goal to reconstruct the explosion from an observed spectral time series of a SN Ia. In most cases this reconstruction starts with a parameterized model of the explosion. In a next step we, manually or algorithmically, adjust the model parameters and use a spectral synthesis code, like TARDIS, to generate theoretical spectra that match the observed ones and then - finally - deduce that these parameters are likely close to the reality (*e.g.*,Si abundance).

Already in the original work by Stehle et al. (2005), which introduced the abundance tomography method, a suitable model for SN 2002bo was identified and numerous other supernovae have been studied with similar success by a number of authors (Hachinger et al., 2009; Mazzali et al., 2015) by relying on a manual execution of the abundance tomography fitting procedure. Recently, Jancauskas et al, in prep. demonstrated that the fitting process can be automated. However, the authors also found that multiple models produce spectra which agree with observations to a similar degree. This degeneracy in some model parameters makes the identification of a best-fit model challenging. In this thesis we intend to algorithmically identify regions of parameter space that produce spectra similar to the observation and examine the degeneracies in the parameter space.

Finding these models requires efficient exploration of the parameter space for which we will use the MCMC algorithm (see section 2.4.2). For each point explored this way, we generate a synthetic spectrum. However running the simulation code is computationally expensive, therefore we use the emulator introduced in section 2.3 to approximate the spectral synthesis process.

In section 3.1 we present our result of using TARDIS directly to generate the spectra needed by MCMC. We found a solution to the computational cost by appropriately interpolating between pre-computed TARDIS spectra on a sparse grid. More specifically, we use an spectral emulator (section 2.3) for this purpose. We show in section 3.2 that this approach does indeed produce spectra that are similar to the synthetic spectra generated by TARDIS.

The spectral emulator has parameters unrelated to physics, such as the number of eigenspectra used by the PCA. We will look at these parameters and optimize them as needed. This includes examining the impact of the training grid on the performance, the data preparation and the interpolation. For the purpose of testing the emulator, we created four different datasets to target specific problems that might arise.

To showcase our method, we need a likelihood function (see section 2.1.3) to quantify the match of a synthetic spectrum and the observation. Construction a specialized likelihood function is out of scope for this thesis, so we combine basic ideas to see, what works best. In section 3.3 we compare our candidates for the likelihood function and select one, that we use for further analysis.

In a final test, we apply the methods developed in this thesis, *i.e.*,the spectral emulator (section 2.3), the likelihood function (section 2.1.3) and the algorithm for sampling the parameter space (section 2.4.2) to fit the supernova SN 2002bo. We will then compare our results with previous studies that are already available (see Stehle et al. (2005)) for this object.

3.1 Markov chain Monte Carlo with TARDIS

As described in section 2.4, we aim to find the best fitting model and associated uncertainties in a means to better understand the explosion. This requires efficient exploration of the parameter space, whose boundaries are listed in table 2.1. We use the MCMC algorithm to generate sample points and TARDIS explicitly to generate spectra for those

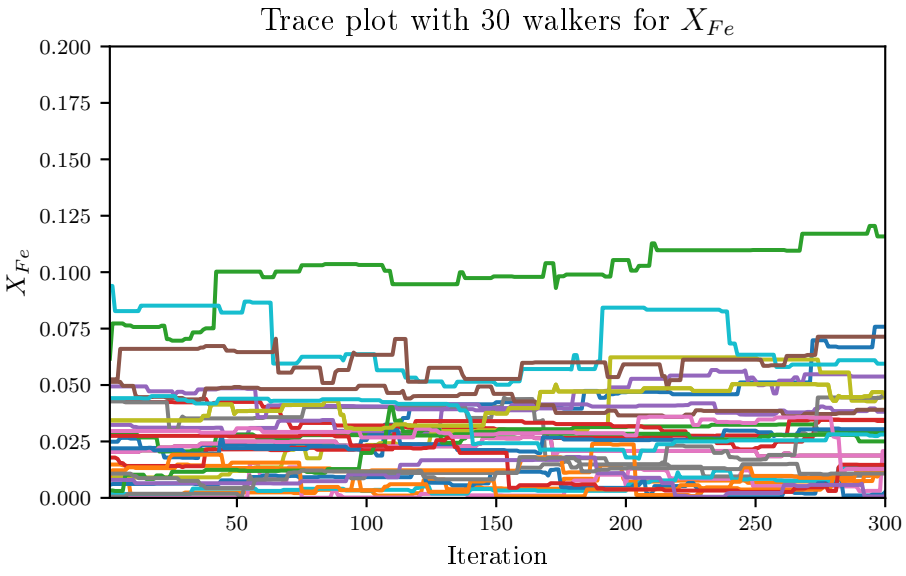

Figure 3.1: This plot shows the iron abundance X_{Fe} of a random subset of 30 walkers after running MCMC for 300 iterations. Important to note is that the walkers do not change values on average and there is almost no mixing. After the burn-in, the walkers are expected to have converged onto an area and started mixing frequently.

points. To quantify the match to the observation, we use the log likelihood

$$L(x) = -\frac{1}{2}\sum_{i=1}^{N}\left(\frac{f_i(\theta) - y_i}{\sigma_i}\right)^2, \tag{3.1}$$

where $f_i(\theta)$ is the value of the i-th bin of the spectrum for a set of model parameters θ, y_i is the i-th bin of the observation and σ_i is the uncertainty in that bin. We restrict this likelihood to the bins between 2800 to 6300 Å.

The spectra created by the simulation code include several uncertainties that are extremely difficult to quantify, such as uncertainties due to atomic data, the one dimension approximation or the Sobolev approximation. Because they have not been studied in detail, we neglected them for now (by setting $\sigma_i = 1$).

We use the Bayesian interpretation of probability (see section 2.1.3) which requires the priors $P(\theta)$ and the evidence $P(x)$. We can neglect the evidence ($P(x) = 1$) because only the model parameters change, not the models. We use priors for the model parameters coming from nucleosythesis calculations and general constraints of the model (see section 2.2.2 for details on priors and table 3.1 for their values).

As the first test we used the above setup and generated samples with EMCEE, an implementation of MCMC. We choose this implementation because it has only two parameters, the number of individual Markov chains, called *walkers*, and a scaling parameter a which we do not change. In contrast to most MCMC implementations based on the Metropolis-Hastings algorithm (see Hastings, 1970) which use a multivariate normal distribution as the proposal function, EMCEE uses the positions of several individual chains to propose new points. The benefit is that the initialization does not require the covariance of the distribution but only positions for each individual chain. Determining this covariance would require a significant number of evaluations to the posterior probability and thereby computational resources due to the time it takes to simulate a spectrum. With EMCEE we expect all chains to converge to the region of parameter space with high posterior probability relatively quickly. We originally expected this burn-in phase to last for approximately 100 iterations, however our analysis in section 3.3 and section 3.4 each result in an autocorrelation time of approximately 100 samples. It is common to ignore at least the first ten independent samples, depending on the problem even more.

We start the exploration of the parameter space by initializing 200 walkers with random points in the twelve-dimensional parameter space that meet our prior requirements (see section 2.2.2), *e.g.,*valid oxygen and an iron abundance of $0 < X_{\mathrm{Fe}} < 0.2$. We propagated the chains 300 times which took a total of 31 h on a total of 512 cores. For 28 643 out of the 6×10^4 proposed points a TARDIS spectrum was generated. For the remaining points no spectral synthesis calculation was performed because the proposed parameter sets were excluded by the imposed priors.

We expected all walkers to converge on the area around the point with maximum posterior probability quickly, *i.e.,*after a burn-in phase of about 100 iterations, and mix there. In

Table 3.1: Parameters and their priors used for the exploration of the parameters with EMCEE.

X_N	Si	S	Ca	Fe	Co	Ni	Mg	Ti	Cr	C	O
min					0						
max	1.0	0.5	0.1	0.1	0.2	0.2	0.1	0.02	0.02	0.1	1

	v_i [km s^{-1}]	T_i [K]
min	5000	6500
max	20 000	20 000

fig. 3.1 we show the trace, that is the history of parameters one Markov chain had, for a subset of the walkers. Clearly there is no convergence after 100 iterations and there is no visible trend which could be used to estimate when convergence will be achieved.

One important aspect of MCMC is the mixing of multiple walkers. While some of the walkers cross their paths, there is no frequent change of parameter which would be expected after the burn-in. This shows that the chains are still at the beginning of the initialization phase, after having already spent a significant amount of computation time. Further tests aimed at improving the initial distribution of walkers by using a distribution gained by fitting a multivariate normal distribution to the likelihood distribution of earlier runs or by using a small spread around the best fitting point found by Jancauskas et al, in prep. produced similar results.

As we discovered later, we made a mathematical mistake that affected the calculation of the likelihood, preventing individual walkers from advancing properly. By setting the uncertainties to $1\,\mathrm{erg\,s^{-1}\,cm^{-2}\,\AA^{-1}}$, we assumed uncertainties orders of magnitude higher than the actual measured values, in our case the emergent flux. The latter is typically of the order of $10^{-22}\,\mathrm{erg\,s^{-1}\,cm^{-2}\,\AA^{-1}}$. Thus, no matter how large the difference in the spectra, the likelihood is dominated by the error and a new point is always accepted.

What should have been done would have been for example to set the uncertainty σ to a value related to the emergent flux. This would mean for example a 5% uncertainty. By accepting every proposed point that was not rejected by the priors we imposed, it was impossible for the chains to converge, explaining the observed behavior. One should note that the method in general works, as we show later with the help of the spectral emulator (see in section 3.4).

Despite this error, this test nevertheless provided valuable insights. By investigating the set of calculated spectra we found that to properly characterize the parameter space many more spectra are required. This in turn calls for a more efficient way of generating spectra if the overall task is to be performed on reasonable time scales. The spectral emulator, which we will study in the next section allows us to repeat this analysis in a fraction of the time.

Table 3.2: Parameters of the emulator with their default values which are used throughout this thesis.

SG filter		PCA	GP			
WL	p	PC	l_{lower}	l_{upper}	nugget	optimizer
21	3	100	10^{-5}	10^{2}	10^{-4}	Welch

3.2 Testing the emulator

In the previous section we showed that the simulation code TARDIS, despite being highly optimized, is too slow for exploring the parameter space with MCMC. In search for a faster way to generate spectra we developed the spectral emulator, a tool that is able to interpolate efficiently between a set of precomputed synthetic spectra. Skipping the expensive simulation of physical processes allows the generation of spectra in milliseconds compared to approximately 100 s with TARDIS. However this approach hides the direct connection between model parameters and their spectra through physical processes by embedding this information in the grid. Therefore it needs to cover the region around interpolation points well. Spectra generated by TARDIS are noisy which may be problematic for MCMC but acceptable for the emulator because it can handle noisy data. Thus, the steady interpolation also solves the problem that MCMC walkers can get stuck on outliers because of noise.

In order to justify substituting the simulation code, it is necessary to test the limitations and capabilities of the emulator. Specifically that means comparing emulator spectra with TARDIS calculations. To this end we use the emulator with four different datasets, specifically created to test its performance in various areas.

Optimizing non-model parameters The emulator itself has parameters, as explained in section 2.3, for example the number of Principal Components (PCs). In table 3.2 we list these, together with their default values. These are a good starting point but need to be carefully checked to ensure an efficient performance and accurate results of the emulator for a specific problem. There are three aspects in which respect the emulator should be optimized, that can be tested and optimized.

Grid dependance The emulator is an advanced approach that allows interpolation on a grid of precomputed spectra. A different dataset will give different interpolation results and the goal is to keep these differences below the level of noise we expect to have in our data. In this case, the noise and systematic errors dominate and we can neglect the interpolation error. We show the variance in the interpolated spectra by using a random subset of the original grid and comparing the spectra for one set of model parameters.

Preprocessing There are two steps for preprocessing the data: Smoothing the spectra and decomposing them into PCs. The smoothing is necessary to remove noise in

the spectra, which the PCA would otherwise identify it as real features. That means the information content does not drop off and almost all components would be required to reconstruct a spectrum from its projection on the PCA basis. We use a SG filter for this, an idea adopted from Sasdelli et al. (2014). The smoothing is done by fitting a p-order polynomial to the w surrounding points and evaluating it in the middle. We optimize these two parameters manually, which is a trade off between loss of information and increased performance of the PCA. The PCA has the number of PCs as parameter. The number of PCs has positive and negative effects on the performance of the emulator. On the one hand, PCs carry all the information so a minimum is required for working spectra reconstruction. However, each PC contains less information than the previous one, slowly getting dominated by noise. Because the interpolation takes time and we do not want to interpolate noise, we should find a good balance.

Interpolation We briefly compare the GP interpolation against the popular Random Forest (see *e.g.,* Breiman, 2001) (RF) regressor. The main topic however is the optimization of GP parameters, which are optimized during the training phase (see section 2.3.3). The GP implementation we use provides two different algorithms for this. We will analyze, whether the improved, but considerably slower algorithm is worth using over the simple one.

Precomputed Grids We created four grids to test different areas of application of the emulator. Except for the one-dimensional grid, we use the same LH (see section 2.3.1) as the source of our grid. This is a grid of values between zero and one that can be transformed onto any interval using a simple transformation. For example to transform $[0, 1]$ onto $[0, \infty)$ one would use

$$f(x) = \frac{1}{1-x} - 1. \tag{3.2}$$

When the datasets are introduced in detail, we will also cover how the LH transforms into the model parameters.

Since the oxygen abundance X_O is computed based on all other parameters (see section 2.2.2), we do not count it as an input parameter, although it is constantly changing. For example in the one-dimensional grid, we change the value of X_{Fe} from 0 to 0.2 and X_O changes accordingly to achieve the normalization to 1 (see eq. (2.7)).

Here is a short overview of all four grids:

- We start with a simple one-dimensional grid with the iron abundance X_{Fe} as the only parameter which changes from 0 to 0.2 (see section 3.2.1). Here we test the general application of the method and do most of the manual optimization regarding the preprocessing of our spectra.

Table 3.3: Parameters used to generate the one-dimensional dataset. They are based on a best fit done by Jancauskas et al, in prep.. Note that X_{Fe} in this case is the abundance used to create the synthetic reference spectrum which is used to showcase the PDF in fig. 2.5.

N	Si	S	Ca	Fe	Co	Ni	Mg
	Ti	Cr	C	O	v_{i} [km s^{-1}]	T_{i} [K]	
X_{N}	0.3972	0.2880	0.005 146	0.023 24	0.002 067	0.035 76	0.036 45
	0.005 971	0.000 375 4	0.003 603	0.2022	13 140	10 820	

- In a second step we increase the complexity and consider a ten-dimensional grid with all abundances varying linearly, to test interpolation in many dimensions.
- In addition to the linear multidimensional grid, we also investigate the influence of logarithmic spacing. With such a spacing we can accurately follow the effect of parameters which vary over orders of magnitude. This would be particularly useful for parameters with low abundance ratios. By using logarithmic intervals for the parameters we hope to thoroughly cover the area needed for interpolation. We repeat the analysis done for the other ten-dimensional grid.
- In a final step and in an attempt to generalize the method, we use a grid with ten abundance parameters with logarithmic intervals together with v_{i} and T_{i} to interpolate spectra for different epochs.

3.2.1 Testing one dimension

We begin the validation process of the spectral emulator by applying it to a simple setup to show that predicting the spectrum for one free parameter is possible. The base model for this test is the best fitting combination of parameters found by Jancauskas et al, in prep. for a fit to a spectrum generated by ARTIS. This model uses the same density profile as discussed in section 2.2.1. We use this template to generate a series of 500 spectra with TARDIS, changing the iron abundance X_{Fe} from 0 to 0.2 Alongside this change comes an adjustment to the oxygen abundance X_{O} to satisfy $\sum_n X_n = 1$ (see discussion in section 2.2.2). We chose iron as the principle parameter for this test because the large number of atomic lines has a noticeable impact on the spectrum. In contrast, oxygen has little impact on the spectrum. That means we can approximate the system as one-dimensional with X_{Fe} as the only parameter. The other abundance parameters, T_{i} and v_{i} have the values listen in table 3.3.

Grid dependence There are two cases we want to test in this section. We show how many spectra are needed to successfully interpolate a spectrum such that the difference to the original is below the level of noise. To this end, we train the emulator with a random subset of N spectra and use the remainder for the comparison. In fig. 3.2 we

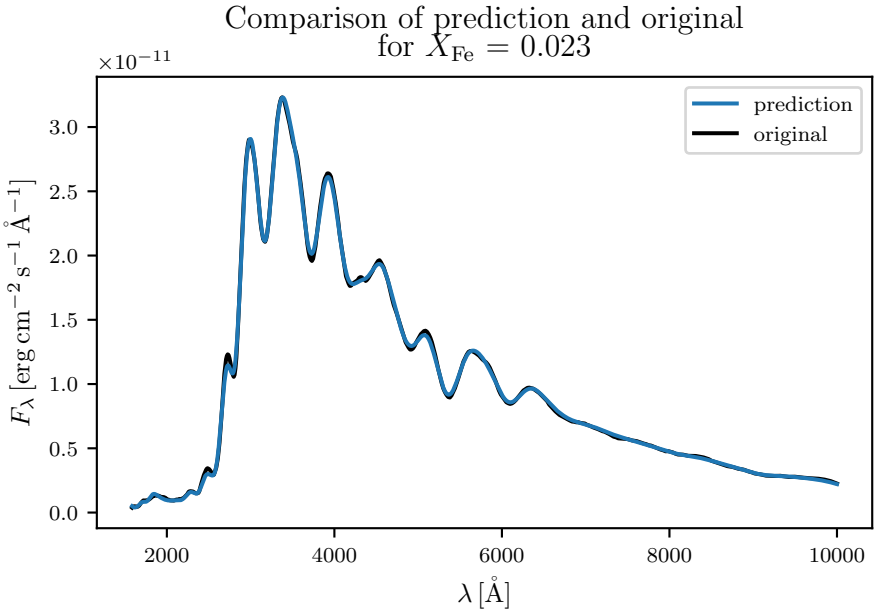

Figure 3.2: Comparison of a synthetic spectrum generated with TARDIS (original) and a spectrum generated by the emulator (prediction). Note that the chosen point $X_{\mathrm{Fe}} = 0.023$ is not part of the training grid and that an artificial offset has been applied to facilitate visual comparison.

show that with a grid of 50 points we are able to reconstruct a spectrum with all its features. It is even possible, to properly reconstruct with as little as 20 points, however this heavily depends on the distribution of the points. However training the emulator with only 20 points often leads to the emulator failing to converge to a fit. In fig. 3.3 we show the result of a prediction for this case. Thus, we will use 50 random points for the remainder of the tests. To highlight the spread of the prediction, we use 50 different subsets to train the emulator with and reconstruct the same spectrum. In fig. 3.4 we present the result of this analysis namely that the spread is low enough to trust the prediction for any random subset.

Preprocessing As discussed in section 2.3.2 we have to prepare the spectra for the interpolation. This is especially important because we optimize the creation of the synthetic spectra with TARDIS towards speed. We do this by choosing a small number of MC packets N to use during the simulation. The noise in the generated spectrum is proportional to $\mathcal{O}(N^{-\frac{1}{2}})$ while the effort to do the calculation scales with $\mathcal{O}(N)$. Thus we have to make a trade-off between speed and accuracy. For our analysis we focus more on performance to keep the already high computational cost of the analysis low. As a result, we choose $N = 4 \times 10^4$ during the plasma convergence iterations and $N = 10^5$ for the final, high-resolution simulation that creates the emergent spectrum.

We then follow the approach by Sasdelli et al. (2014) and use a SG filter to reduce the noise and provide smooth spectra that can be used by the PCA. This filter has two parameters, the window length WL and the order of the polynomial p that is fit to the points inside the window. We solve this optimization problem manually. In fig. 3.5 we compare different parameters for the SG filter. The plot shows the trend of a higher order polynomial being able to represent the data better. Increasing the window length increases the smoothing potential of the filter. At the end of the day, we have to make a trade-off between retaining sufficient information and facilitating the fit which requires smooth data. For this reason we choose to use a window length of 21 points, that means ten points on either side of the current location are considered for the fit. To represent the data, we use a third order polynomial. Although we lose a lot of information that way, we had to choose these settings for the whole framework to function properly. The current implementation is sensitive to the training data and these settings for the SG filter help improve the robustness of the emulator. A better choice would be a fifth order polynomial and a shorter window length, around 15, which would represent the date much better and not remove small peaks (see around 4300 Å in fig. 3.5).

We determine the number of principal components we need for this grid by plotting the associated coefficients. In fig. 3.6 we show the first three components. The forth and further components have coefficients that are zero almost all the time. We can thus truncate the series after three components without losing much information for this one-dimensional test.

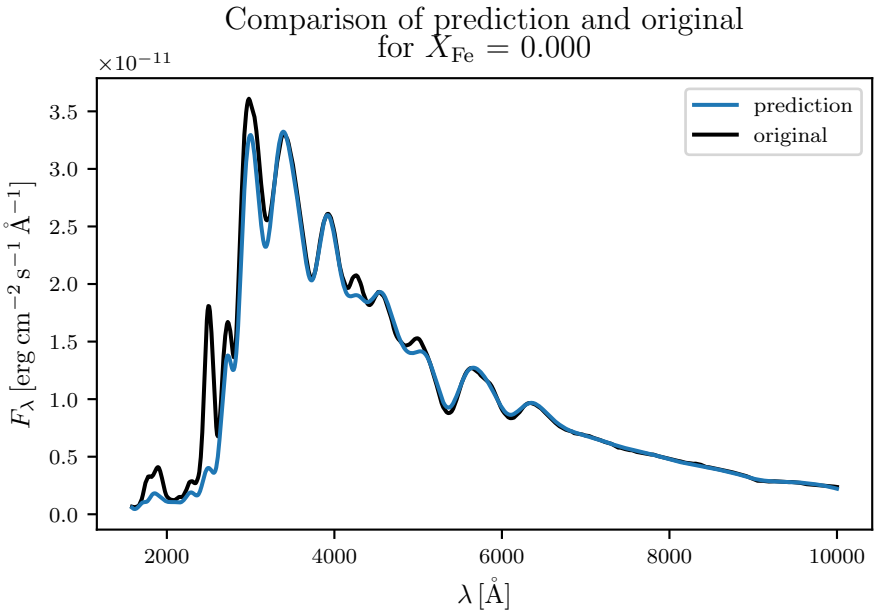

Figure 3.3: Same as fig. 3.2 but we only use 20 points for the interpolation. There are clearly features missing below 4000 Å. Note that to better highlight the difference, we are not using an offset for the plot.

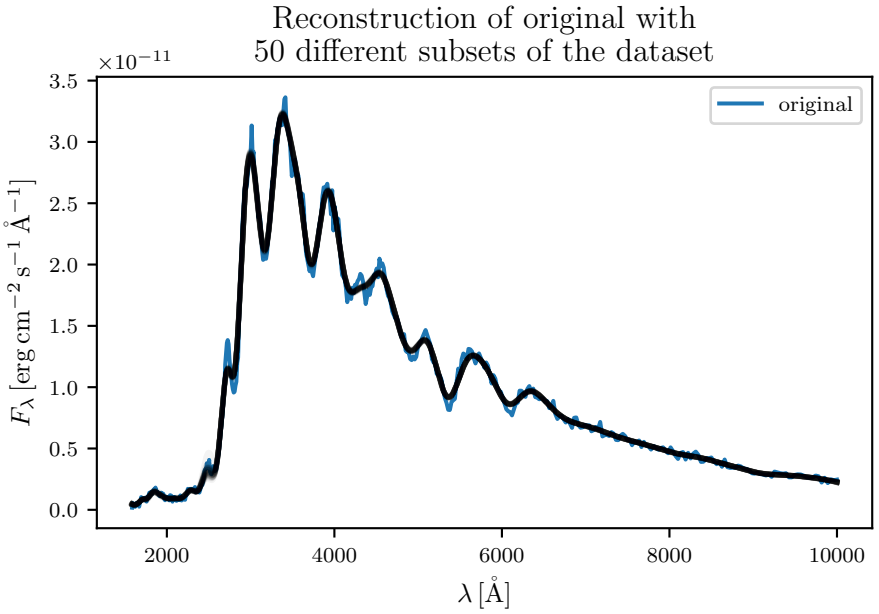

Figure 3.4: We show the spectrum the emulator predicts for the one-dimensional case for 50 different subsets containing 50 points out of the 500 datapoints. These different spectra are plotted in black with a transparency of 10 %. That means areas where all spectra overlap are darker while areas with spread are lighter. Overplotted in blue is the original spectrum calculated with TARDIS.

Interpolation We use a GP for interpolation. This has the advantage, that it provides an error estimate for the interpolation process. That is, interpolation in an area with a low density of reference points results is more uncertain and thus results in a higher error estimate than high density regions. However the time the GP takes to train and predict scales unfavourably with the number of dimensions, principal components and points in the grid (see fig. 2.4). As an example, training the emulator with a grid of 1000 points takes 12 s, while 2000 points need 48 s. This is a scaling of exactly $\mathcal{O}(N^2)$ which confirms the estimate from section 2.3.3. The requirements to the computing hardware scale the same, especially the required Random Access Memory (RAM) reaches amounts only available on computing clusters if the number of points exceeds 10 000.

The GP is able to interpolate the data well with the default values mentioned in section 2.3. For the one-dimensional application the advanced optimization algorithm is not available. This leaves the boundaries for the length-scale l_{Fe} as the only variable parameter. For the upper boundary we choose a value of 10^2 because anything above this value would correspond to a flat distribution which would be the result of PCs that could be ignored. For the lower boundary we should choose a value that is bigger than the distance between two points in the grid. Otherwise it might be possible to end up with the interpolation only working in the vicinity of grid points and failing in between. We choose a value of 10^{-5} which works well for this grid. With the emulator trained

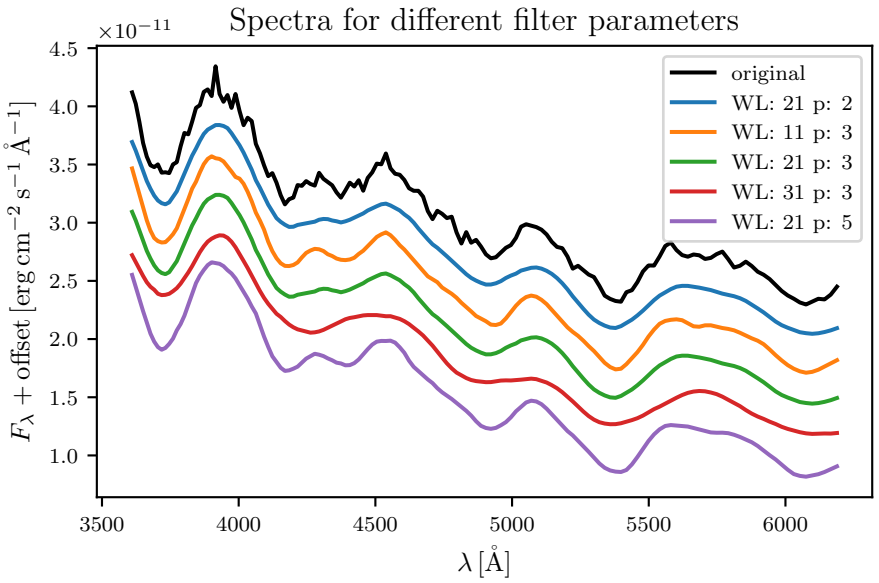

Figure 3.5: We show different combinations of parameters for the Savitzky-Golay (Savitzky & Golay, 1964) filter (SG filter). We denote the window length by 'WL' and the oder of the polynomial by 'p'. There are two trends in the data: Increasing 'p' increases the sharpness of the filtered spectrum, allowing for narrow peaks to appear in the data. On the other hand, increasing 'WL' creates a smoother spectrum by including more points. For the remainder of this work, we choose 'WL'= 21 and 'p'= 3.

with these numbers we are able to reconstruct spectra that match a synthetic spectrum generated by TARDIS well (see fig. 3.2). There are only small features missing that could as well be noise in the original data.

In search for a possible replacement to the GP we try the popular RF classifier. The big advantage of this algorithm is that the evaluation does not scale with the number of data-points used for training. Additionally the training for a huge dataset takes a fraction of the time needed by the GP. However we were not able to create spectra resembling the original with this approach and therefore we did not pursue this method for the current application further. We therefore use the GP interpolation for the remainder of this work.

3.2.2 Abundance test

With the emulator working well in one dimension, the next step is to allow all abundances to vary. We can not use the approach from section 3.2.1 to create a grid with ten dimension as the simulation thereof would take prohibitively long. Instead we randomly spread 7×10^4 points across the parameter space creating a LH (see section 2.3.1). We

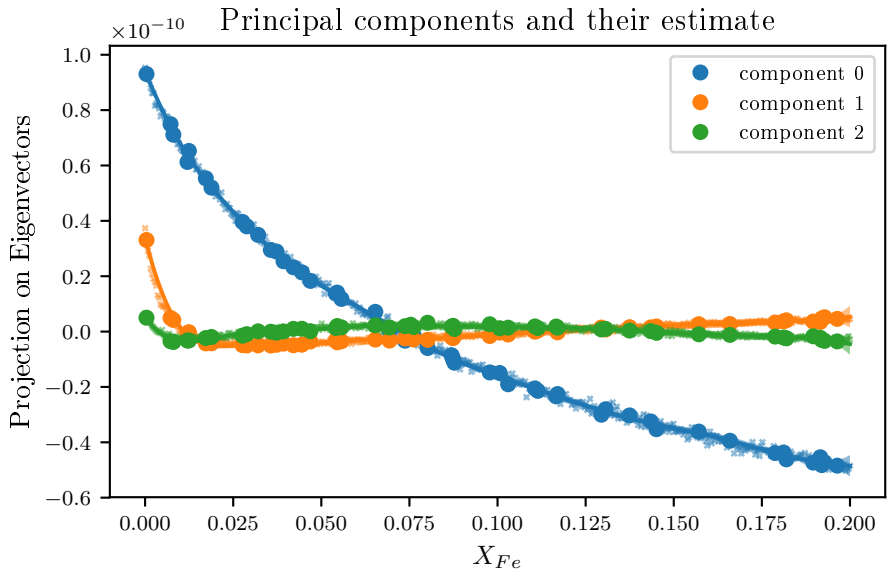

Figure 3.6: Illustration of the successful interpolation of the emulator in the PCA space for the one-dimensional iron test. Only the first three PC are shown because higher other coefficients do not contain information (*i.e.*,they are almost zero). The points show the data-points that are used to train the emulator. The crosses are all data-points available in the dataset.The solid line which shows the prediction of the emulator falls nicely on these points, thus demonstrating the successful performance of the emulator. The light background shows the uncertainty associated with the prediction.

transform these points into model parameters using the boundaries in table 3.4. For v_i and T_i we choose $11\,700\,\mathrm{km\,s^{-1}}$ and $10\,000\,\mathrm{K}$ respectively. Because the LH covers the whole parameter space without respecting the priors we defined earlier (see section 2.2.2), we filter out all points that do not meet our requirements. This leaves us with approximately 10 000 points for which we create a spectrum using TARDIS. This means that the priors we use filter out roughly 6/7 of the parameter space if we use the linear map (eq. (3.3)).

We will go into detail why we created two different grids down below. The difference between them is the way the intervals for each abundance parameter are spaced. The first does linear interpolation between 0 and the upper boundary while the second set interpolates logarithmically between a lower boundary and the upper boundary. For both sets the oxygen ratio is calculated afterwards. Because of our prior requirements for the parameters a big part of the 7×10^4 points is excluded by the priors, netting approximately 1.05×10^4 points for the linear interpolation and 3.65×10^4 for the logarithmic interpolation. It is expected to get more data for the logarithmic interpolation because more points lie near the lower end of the interval.

Table 3.4: Parameters for the two different LHs used in section 3.2.2.

Name	10d linear		10d log	
	min	max	min	max
Si	0	1.0	10^{-3}	1.0
S	0	0.5	10^{-3}	0.5
Ca	0	0.1	10^{-5}	0.1
Fe	0	0.2	10^{-5}	0.1
Co	0	0.2	10^{-5}	0.2
Ni	0	0.2	10^{-5}	0.2
Mg	0	0.1	10^{-5}	0.1
Ti	0	0.02	10^{-7}	0.02
Cr	0	0.02	10^{-7}	0.02
C	0	0.1	10^{-5}	0.1

Linear grid We present the results of using a grid with 1.05×10^4 points created from a LH with 7×10^4 points and transforming its values from $[0, 1]$ onto the ranges shown in table 3.4 using eq. (3.3). As mentioned earlier, we keep v_i=11 700 km s^{-1} and T_i= 10 000 K fixed. To transform the uniform values of the LH into the parameter values we use

$$f(x) = l + (u - l)x \tag{3.3}$$

with the lower l and upper u bounds and the value $x \in [0, 1]$ from the LH.

Going from one to ten model parameters increases the variance in the spectra. We cover the full range for each abundance parameter, from no influence to the spectrum until the atomic lines are saturated. We need approximately 100 PCs to create spectra similar to the originals (see fig. 3.7).

We use the default values for the emulator parameters shown in section 2.3 and repeat our approach from section 3.2.1: We train the emulator with a subset of the original grid (5×10^3 points) and use the remainder to validate the interpolation results by comparing them to the original.

The result works well, however we notice the interpolation does not work near the edges of the training grid (see fig. 3.8). Specifically, if one is interested in the iron abundance X_Fe between 0.001 to 0.01 Å the whole grid has only 651 points in this area. However, adding another requirement, like $X_\mathrm{C} < 0.01$ drops this number to 74. As a result interpolation and extrapolation mix and the resulting spectrum does not always resemble the original. In fig. 3.8 we compare interpolation for a point near the center of the training grid (P1) and a point at the edges (P2). The model parameters for these points can be found in table 3.5.

Logarithmic grid We also explore other ways of mapping the LH to the model parameters, in particular by using a logarithmic transformation. In doing so, we increase the

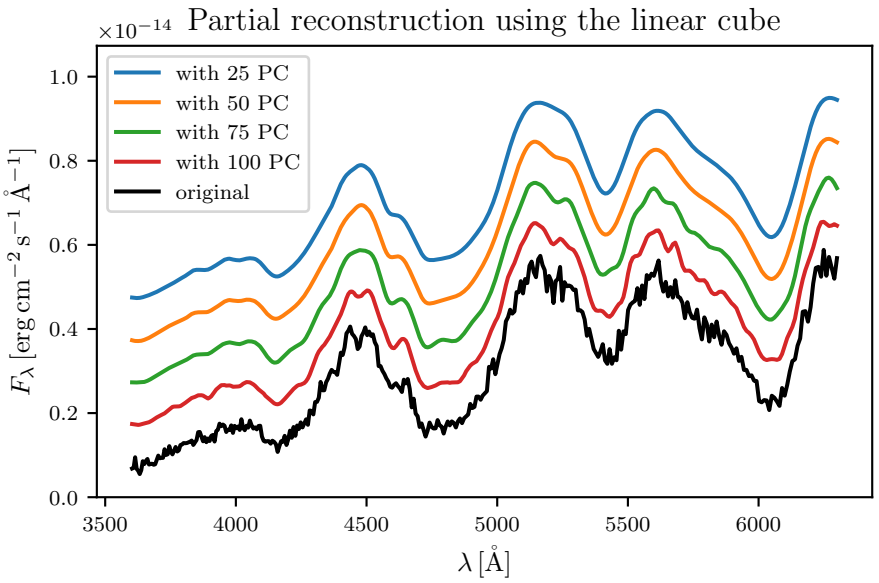

Figure 3.7: Predicting a spectrum with subsequently more PCs. Only using 100 PCs adds detailed features like the double peak at 4500 Å. The parameters corresponding to this spectrum are listed in table 3.5 as P1.

grid density for low values while decreasing it for higher values. We motivate this by assuming that small changes in high abundance ratio regions change the spectrum only slightly while for example changing the iron abundance X_{Fe} from 0.1 to 0.2 completely changes the absorption profile (see fig. 3.9).

For the mapping we use

$$f(x) = u^x \cdot l^{1-x} \tag{3.4}$$

to map the value x between 0 and 1 onto the range between the lower boundary l and the upper boundary u. Overall using this grid, it should be possible to successfully emulate spectra also in regions where there were only a handful of reference points available in the linear cube. Repeating the test from section 3.2.2 with this dataset shows that the issue of extrapolation is solved (see fig. 3.10). Using 10^4 points from this grid it is possible to emulate spectra in a bigger region of the parameter space. One could use all 3.65×10^4 points to train the emulator, however because of the scaling of the requirements as well as the computational effort, we choose 10^4 as a good balance. Therefore we adopt the logarithmic spacing in the parameter grid for any further applications. Nevertheless it is always important to compare prediction and original for a small set of data to justify using the emulator.

Preprocessing For the SG filter we copy the parameters from section 3.2.1 because the best values depend on the shape of the spectra, which is similar. With more variance

Table 3.5: Parameters of the points used to compare the linear and logarithmic transformation of the LH.

	P1	P2
Si	3.1×10^{-1}	4.5×10^{-1}
S	1.6×10^{-1}	1.6×10^{-1}
Ca	7.6×10^{-2}	1.4×10^{-2}
F	8.4×10^{-2}	1.2×10^{-3}
Co	9.0×10^{-2}	1.2×10^{-1}
Ni	8.9×10^{-2}	2.9×10^{-4}
Mg	3.6×10^{-3}	2.2×10^{-3}
Ti	1.2×10^{-2}	3.8×10^{-7}
Cr	1.3×10^{-2}	1.3×10^{-2}
C	3.9×10^{-2}	2.8×10^{-2}
O	1.2×10^{-1}	2.2×10^{-1}

in the spectra, the PCA needs more PCs to achieve an accurate representation. As mentioned earlier in section 3.2, the optimization of the number of PCs is a compromise between speed and accuracy. For this grid, we can again use 100 PCs to reconstruct the spectra (see fig. 3.11).

Interpolation In multiple dimensions the parameters of the GP are more difficult to tune. The area that is influenced by each grid point, is characterized by a length-scale l_N. For this grid we choose a ten-dimensional kernel function (see eq. (2.15)) because the influence of each input dimension (abundance parameter) stretches across a different area. Each length-scale l_N is thereby associated with one input dimension.

The optimization algorithm, which finds suitable length-scales l_N, has the biggest influence on the quality of the interpolation. For one dimension l is a single value and optimization is done by a least squares. However, in multi dimensions l_N becomes a vector and one has to choose whether to optimize all components simultaneously or individually. The latter is computationally more expensive during the training phase but does not affect the evaluation time. Our results show the improvement is noticeable, thus we will adopt this approach for the remainder of this work.

Independent of the algorithm, it is possible for the optimization to fail and the best values for some, or all, l_N coincide with the upper or lower boundaries. A possible explanation is that we use one GP to describe all coefficients of the PCA instead of one for each. This approximation allows us to run the analysis orders of magnitude faster and with considerably reduced requirements in terms of computational resources. This is a trade-off we made that could be solved in a future analysis.

In section 3.2.1 we briefly discuss our attempts with RF interpolation, which were unsuc-

cessful in one dimension. For completeness, we repeated the test for our ten-dimensional grid but obtained the same, discouraging, results.

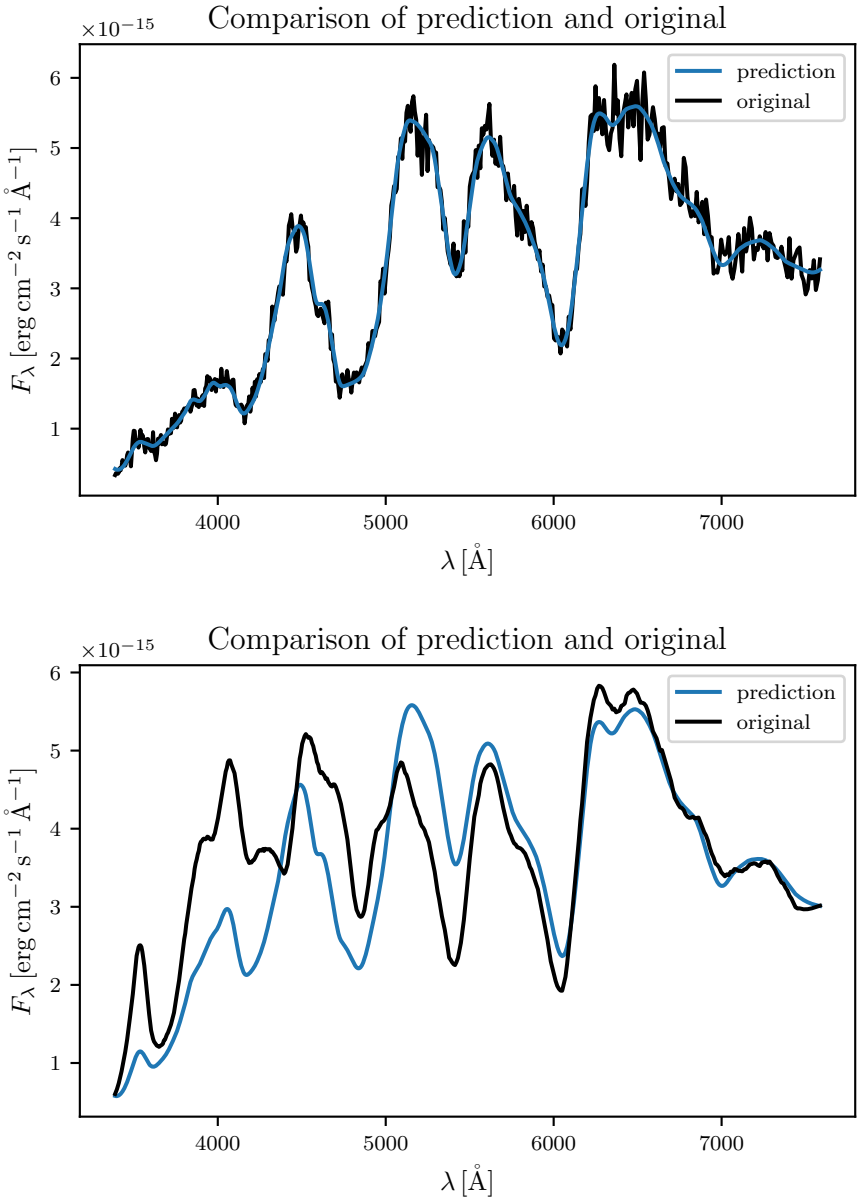

Figure 3.8: Comparison of emulator prediction (prediction) and synthetic spectrum calculated with TARDIS (original) for two different points in the parameter space. The top is a successful prediction while the prediction for the bottom spectrum is wrong because the linear cube does not have enough grid points in that area. The parameters corresponding to these spectra are listed in table 3.5 as P1 (top) and P2 (bottom).

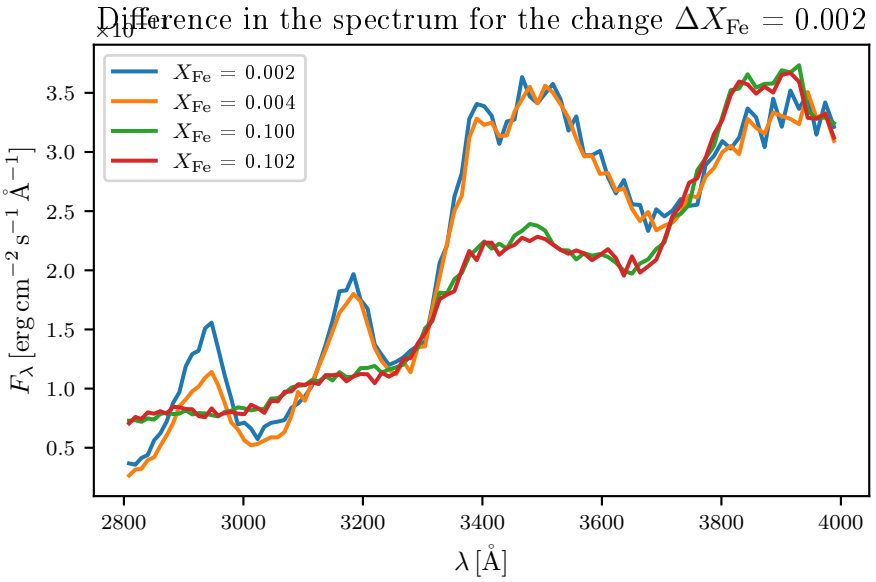

Figure 3.9: We show the difference in a spectrum for a small change in the iron abundance ($\Delta X_{\mathrm{Fe}} = 0.002$). This change is once done for near zero and once for 10 % iron. The plot shows that this change has a notable impact on the spectrum near an abundance of zero while there is no visible difference near 10 %. These spectra were generated with the setup used in section 3.2.1.

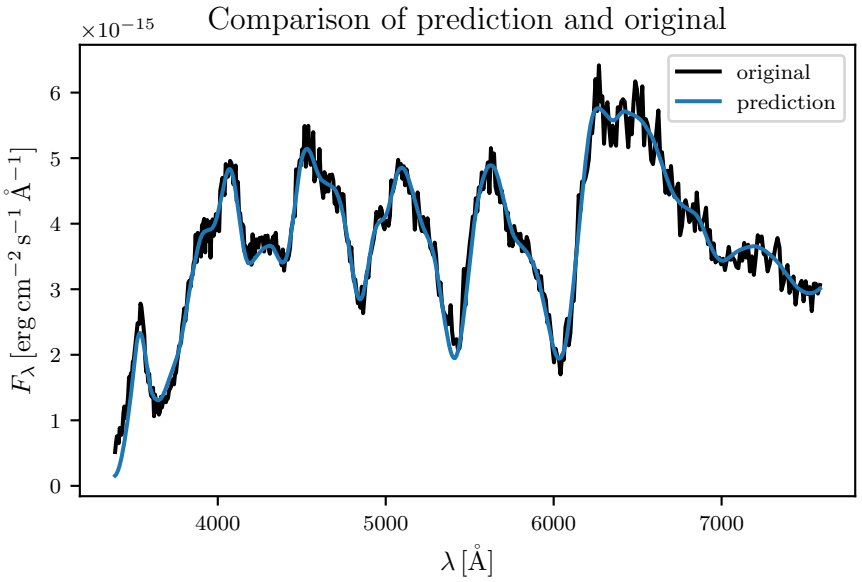

Figure 3.10: Same as the bottom plot in fig. 3.8. However, this time we use the logarithmic transformation of the values of the LH. With this dataset, there is no extrapolation and the spectra match well. The abundance parameters used for this comparison are listed as P2 in table 3.5.

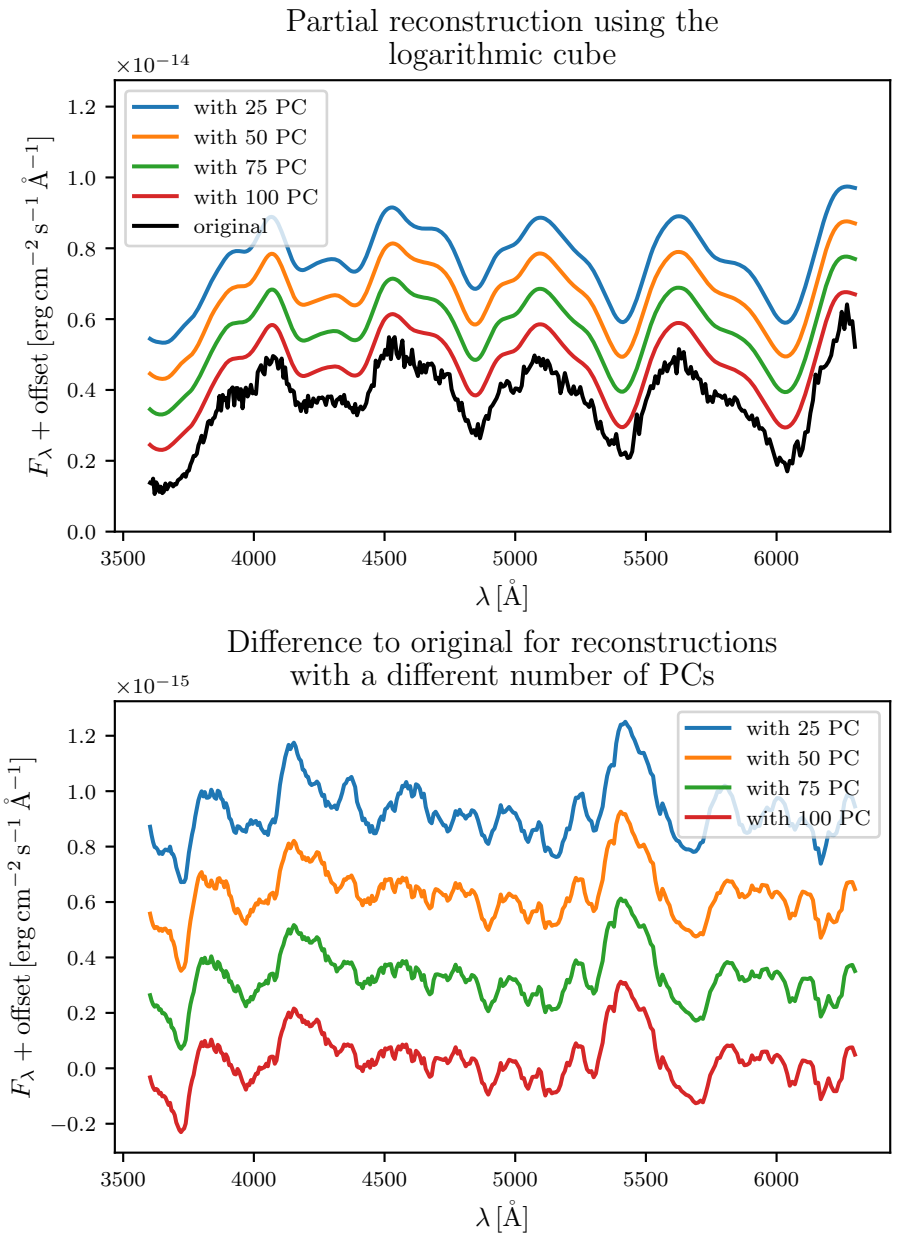

Figure 3.11: The top plot shows the reconstruction of a prediction spectrum with subsequently more PCs. In the bottom the remaining difference is shown between the prediction and the synthetic spectrum from TARDIS. Note that the y-Axis of the latter plot has different limits. The bottom plot highlights how the PCs affect the details of the spectrum (see *e.g.*,at 4400 Å and 5900 Å).

Table 3.6: Boundaries used for the transformation of the LH to model parameters to create the twelve-dimensional grid.

	min	max
Si	10^{-3}	1.000
S	10^{-3}	0.5000
Ca	10^{-5}	0.1000
Fe	10^{-5}	0.1000
Co	10^{-5}	0.2000
Ni	10^{-5}	0.2000
Mg	10^{-5}	0.1000
Ti	10^{-7}	0.020 00
Cr	10^{-7}	0.020 00
C	10^{-5}	0.1000
O	0	1
v_i [km s^{-1}]	5000	20 000
T_i [K]	6500	20 000

3.2.3 Full test

Up to this point, we only varied the composition but kept T_i and v_i constant. However, when fitting actual observed spectra, in particular for multiple epochs, these values also have to become part of our parameter set which is varied. Thus, as a final test, we extend the grid from the previous section by these two parameters and create the fourth dataset which includes all model parameters from the logarithmic grid (see section 3.2.2) and adds T_i and v_i as variables with linear scaling (see eq. (3.3)). The boundaries are listed in table 3.6.

Preprocessing the data in this grid turns out to be a challenge. Because of the nonlinear effects T_i and v_i have on the spectra we have to take special care during this step. Changing T_i changes the integral of the spectrum and moves the position of the peak of the black body background according to Wien's displacement law (see *e.g.,* Heald, 2003). This is a problem for the following PCA which works best with data normalized to a standard deviation of 1. We do not normalize the data to preserve as much information as possible and reversing the normalization for interpolated spectra would be a difficult task. For the previous analysis, we could skip this step because the standard deviation does not differ by orders of magnitude.

The effect of v_i on the spectrum is also problematic, although not as strong as for T_i. The line profile changes with v_i which again is a problem for the PCA. Features, that change in shape instead of amplitude are not clearly associated with single PCs. However such a one-to-one relationship is needed for an accurate and efficient performance of the interpolation. In fig. 3.12 we show two sample predictions that highlight the problems

caused by T_i. The interpolation is not able to predict spectra that match the total luminosity of their synthetic counterparts.

Overcoming these problems requires more research and careful preparing of the data which goes beyond the scope of this thesis. Potential solutions to the interpolation challenge may involve the following steps:

- We can normalize all spectra and thus compare the shape and ignore the integral of the spectrum which corresponds to the total luminosity emitted by the SN.
- Another promising approach may be to reduce the parameter ranges drastically (for example from 5000 to 20 000 to 9000 to 11 000). Spectra would overall have similar shape due to v_i and T_i, however the problem remains that features created by these parameters do not easily translate into the PCs we use to interpolate. The disadvantage of this method is that, again, one needs to create one grid per observation. This leaves the estimate of the ranges for T_i and v_i as the only benefit of this method.
- Other methods of dimensionality reduction and feature extraction beside PCA, might allow us to work with the variety of spectra. For example, Sasdelli et al. (2016) developed a method that uses ML techniques to define a feature space in which SNe Ia can be classified. A similar approach could be used to define such a space for interpolation.

However all of these proposals have not yet been carefully tested, a task which is reserved for future work.

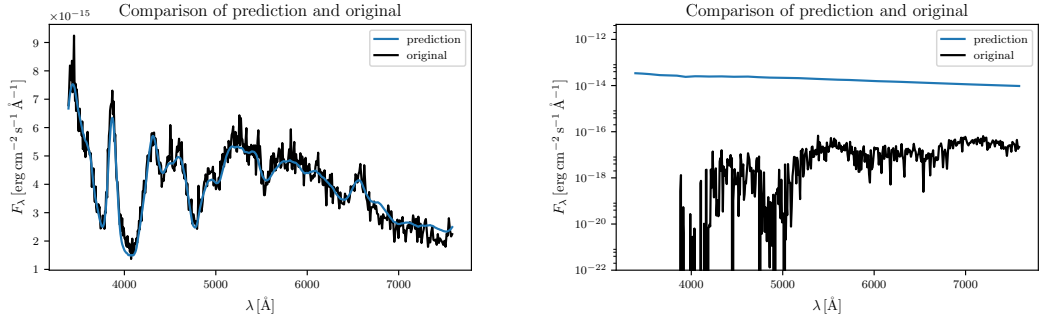

Figure 3.12: Comparison of the emulator prediction (prediction) and a synthetic spectrum (original) for the same point generated with TARDIS. On the left the prediction works well, however on the right, the prediction is completely off, predicting at least an order of magnitude higher luminosity than the synthetic spectrum. The synthetic spectrum was generated with T_i= 5052 K which is considerably less than most other spectra and as a result the PCA does not cover that part well. This highlights the strong influence the T_i parameter has on the luminosity of the spectrum.

3.3 Comparing likelihoods

As discussed in section 2.1 it is difficult to assess the quality of a fit to an observed supernova. We established that automating the exploration of the parameter space needs a metric to compare spectra with. Because we have prior knowledge about the system it is convenient to choose the Bayesian interpretation of probability and the posterior probability as the metric. The posterior probability consists of the prior probability (see section 2.2.2) and a likelihood function, which determines the similarity between observation and model spectrum. We had the goal to find a simple likelihood function that performs well in finding a spectrum that is close to the observation.

To this end we compare four different implementations of a likelihood function against a synthetic spectrum generated with TARDIS. That way we can tell exactly how good each implementation performs because we know the exact model parameters our mock observation was created with.

We train the emulator with the same dataset that we used in section 3.2.2 except for $v_\mathrm{i} = 13\,135.3\,\mathrm{km\,s^{-1}}$ and $T_\mathrm{i} = 10\,821.58\,\mathrm{K}$. Due to the problems outlined in section 3.2.2 we choose to fit a synthetic spectrum in an area that is well covered by the training data.

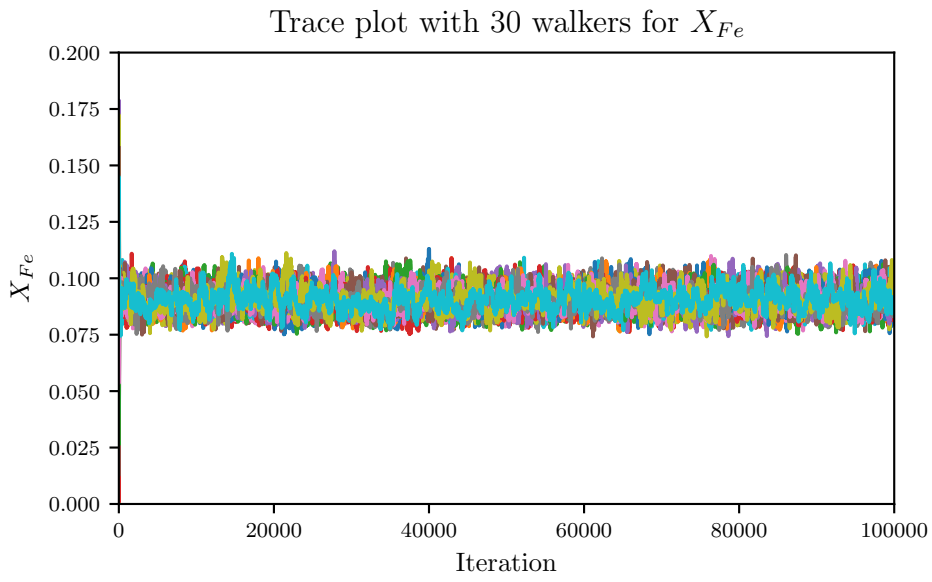

Figure 3.13: We show the iron abundance for a subset of 30 walkers obtained with the basic χ^2 likelihood for a synthetic spectrum as the 'observation'. The constant mixing of the walkers shows that they have reached an equilibrium state and we sample from the posterior PDF.

We limit all likelihood functions to the range from 3600 to 6300 Å because we expect the least unquantifiable uncertainties of TARDIS in this region. Further, when talking about likelihood functions in this section, we mean the logarithmic likelihood $L = \log P(x|\theta)$ that is numerically easier to work with.

For each likelihood we show the parameter surface obtained after using the MCMC algorithm with 300 walkers for 10^5 iterations with a random distribution at the start. For comparable results, we use the same trained emulator and the same initial conditions for EMCEE. We discard the first 10^4 samples to ensure the Markov chains reached equilibrium. From the remaining samples we calculate an autocorrelation time, that is the number of iterations it takes for two samples to be independent of each other. Subsequently we filter the data such that only independent samples remain. For the basic χ^2 likelihood we show exemplary that the Markov chains converged to an equilibrium state (see fig. 3.13). To compare the performance of the individual likelihoods, we plot the approximation to the posterior PDF.

Basic χ^2 ('default') Our first attempt for a likelihood function is assuming a Gaussian distribution for the data. Expanding this for multiple data-points yields a likelihood resembling a χ^2 (see eq. (2.1))

$$L(x) = -0.5 \cdot \sum_i \left[\left(\frac{g_i(x_i) - y_i}{\sigma_i} \right)^2 + \log 2\pi\sigma_i^2 \right]. \tag{3.5}$$

If σ does not depend on x, one can neglect the second term and one ends up with $L(x) = -0.5\chi^2$. However, in section 2.1.3 we introduced the parameter γ that is necessary to model the unknown uncertainties of TARDIS. This parameter is part of x so we have to use the formula from eq. (3.5) with the uncertainty

$$\sigma^2 = ((0.05 \cdot \bar{y})^2 + g_i(x)^2 \cdot \exp(2\gamma)), \tag{3.6}$$

where γ appears as a nuisance parameter that adjusts σ such that 66% of the points are within the errors (see Hogg et al., 2010). During the analysis we will marginalize over γ.

In fig. 3.14 we present an extract from the full PDF associated with this likelihood. All parameters are recovered well and there is only a little spread. In fig. 3.18 we show how the remaining parameters interact. Most parameters match the reference with the exception of Titanium and Chromium where the reference value was not recovered. Additionally, the constraints for Magnesium are broad. Overall, this likelihood is able to reproduce the reference parameters well enough and seems as a good basic approach because it tries to minimize the total error by optimizing the spectrum to fit the underground, rather than individual lines.

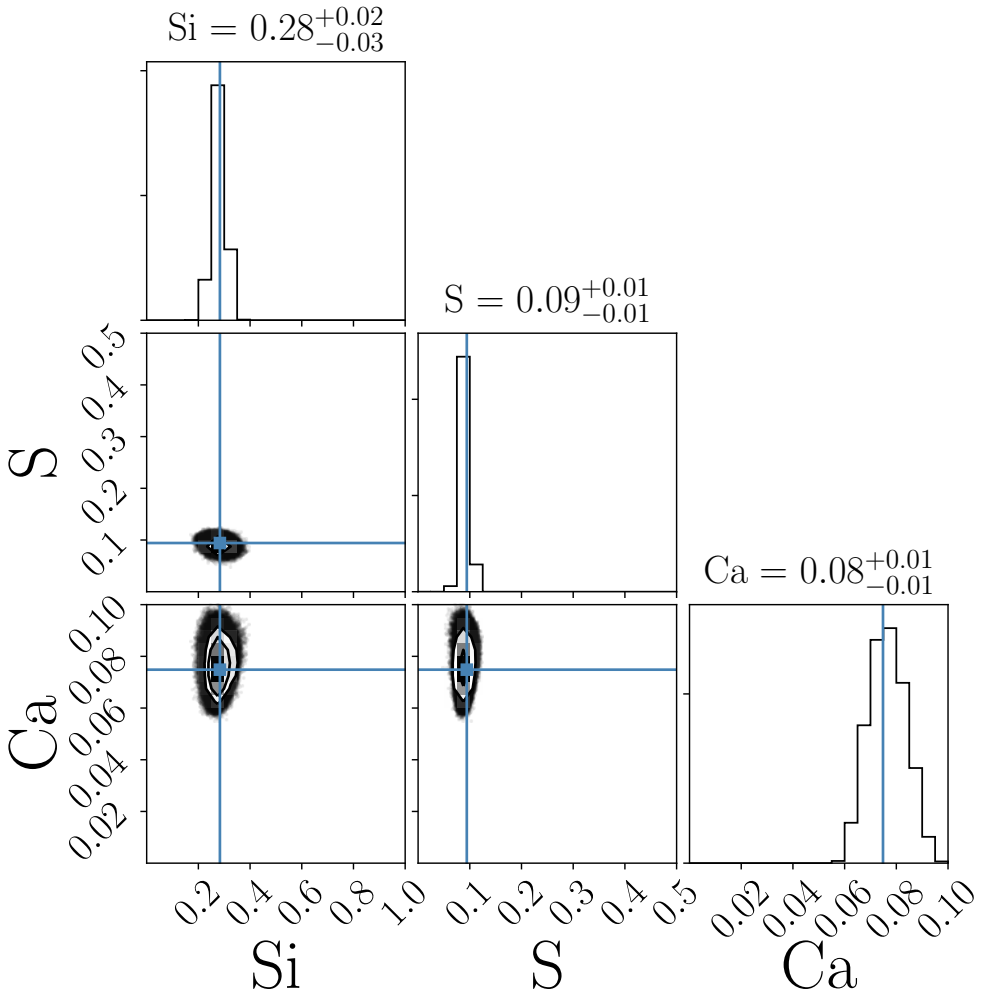

Figure 3.14: Projection of the PDF onto the Si-S, Si-Ca and S-Ca plane to visualize the distribution of the 'default' likelihood. The blue lines indicate parameters of the model used to create the observed spectrum. The parameters shown here are recovered well. For a detailed look at all parameters see fig. 3.18.

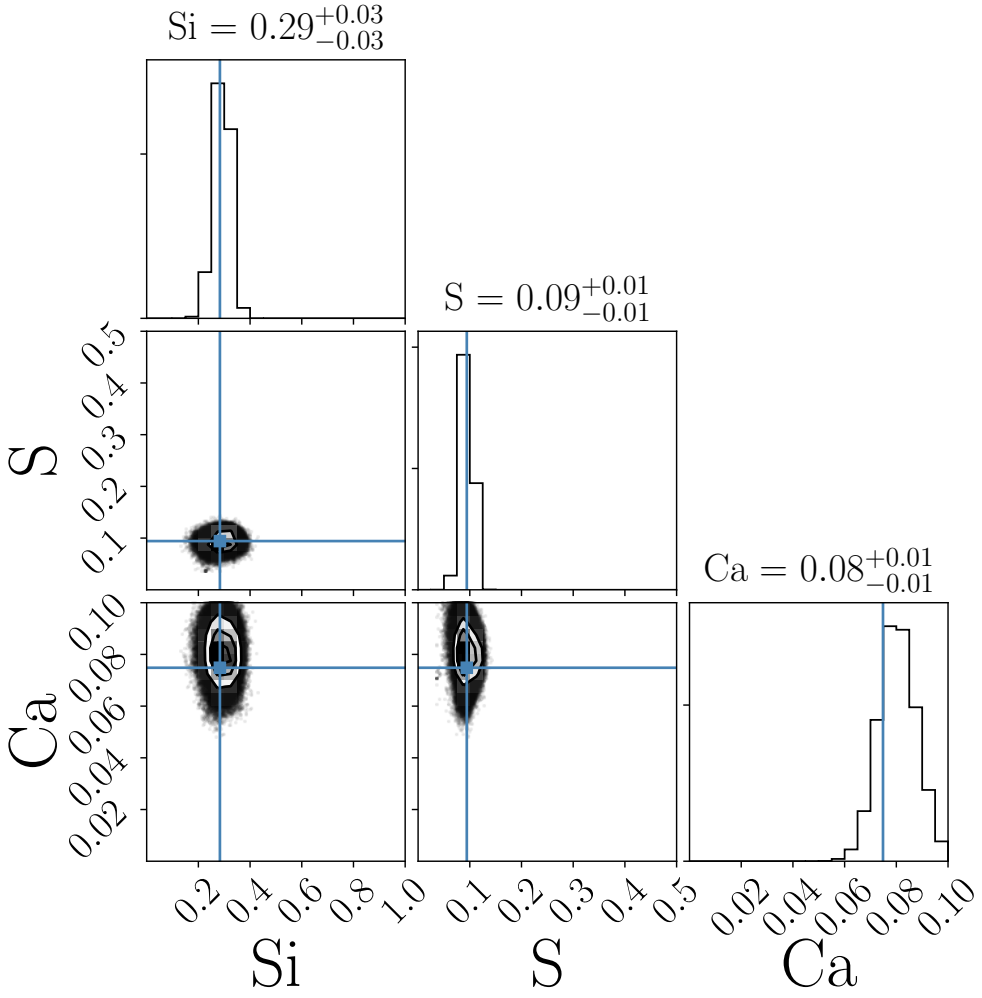

Figure 3.15: Same as fig. 3.14 we compare the spectra with the 'poly' likelihood, where we subtract a fifth order polynomial prior to calculating a χ^2. All parameters shown here match the reference well. For a detailed analysis see fig. 3.19.

χ^2 - subtract continuum ('poly') The main problem of the basic χ^2 is that lines are not given more weight to although they are the features that define a spectrum and therefore the quality of the fit. To overcome this issue, one approach is to subtract a background. Thus, both $g(x)$ and y are preprocessed by fitting a fifth order polynomial to the data and then subtracting it. The result is a spectrum with a mean close to zero where lines translate into strong features. By using the spectrum directly and only subtracting a background, which is determined for each spectrum individually, we are sensible to lines in general and to their depth in particular. However, this likelihood might have the problem that a constant offset of the model spectrum, for example due do high absorption or a low temperature of the photosphere, goes unnoticed. As a result special care should taken when using this likelihood.

We show the results we obtain using this likelihood in fig. 3.15 and fig. 3.19. This likelihood performs almost as good as the basic χ^2 in this scenario, as the most probable point for most parameters agrees well with the reference. Notable outliers are Cobalt, Nickel and especially Chromium, which does not match. In summary, this likelihood could be a contender to the basic χ^2 likelihood and it is worth testing it for fitting a real SN.

χ^2 with differentiation ('diff') Inspired by Sasdelli et al. (2014) who use the differentiation of the spectra for the analysis, we try to find out whether this approach can be used to construct a useful likelihood for SN Ia. The effect is similar to the previously presented method, in that we have an increased focus on the lines. The difference, however, is that this likelihood tries to match the slope of the lines instead of the depth. As in the previous likelihood, we preprocessing $g(x)$ and y individually using the same algorithm. We calculate σ with eq. (3.6) before we do the preprocessing. In fig. 3.16 we show the PDF we obtain when using this likelihood. Only Silicon is well reproduced and in general this likelihood seems not promising without further modifications.

L1 norm instead of L2 ('abs') This likelihood is different from the others because it is based on the L1-norm ($|x|$) instead of the L2 ($\sqrt{x^2}$). This translates to the assumption that the observations follow a Laplace distribution instead of a normal distribution. The general effect is, that regions with a big difference between observation and model do not have a reduced impact on the likelihood. We use σ from eq. (3.6) because the same assumptions about the uncertainties apply. We calculate the likelihood as

$$L(x) = -\sum_i \left| \frac{g_i(x_i) - y_i}{\sigma_i} \right| + \log 2\sigma_i. \tag{3.7}$$

In practice this likelihood behaves similar to the basic χ^2 with the only major differences being the Mg abundance (see fig. 3.17).

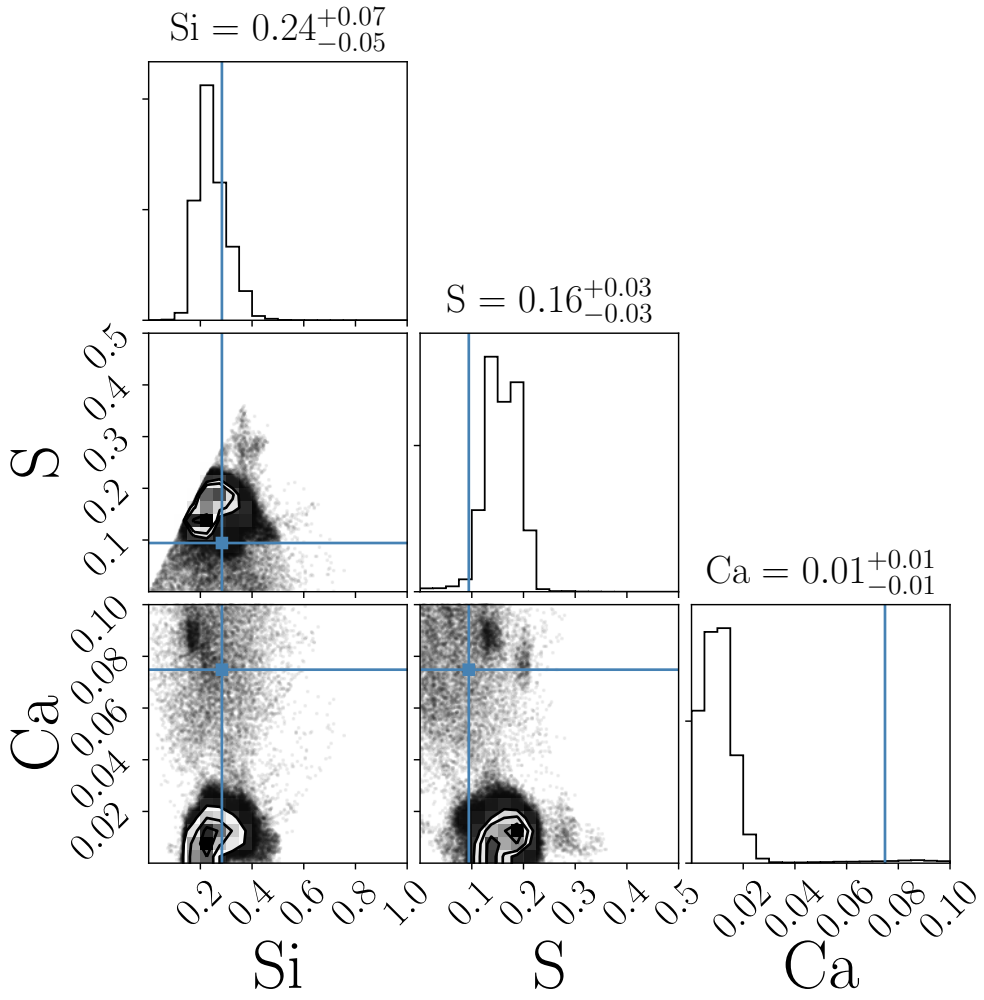

Figure 3.16: Same as fig. 3.14 but we use the differentiation of the spectrum for the χ^2 likelihood ('diff'). The Silicon abundances matches the reference value relatively well. In contrast, the Sulfur is not well determined and the Calcium abundance is even worse. For this likelihood to be usable, some changes would be required.

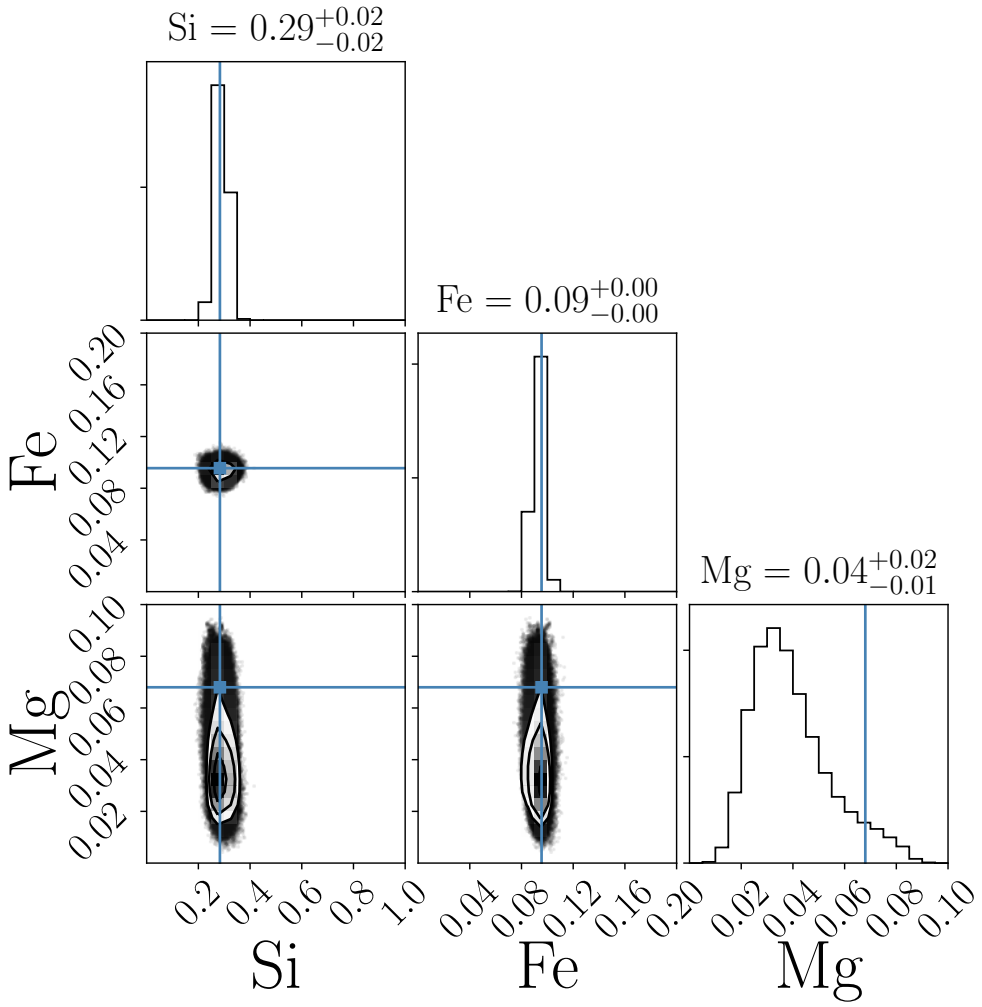

Figure 3.17: Same as fig. 3.14 but we use the Laplace distribution instead of a normal distribution as the basis for the likelihood ('abs'). Note that this plot shows the Si, Fe and Mg abundances to highlight the areas where this likelihood performs different to the χ^2 as their results are generally similar. The important difference is that the 'abs'-likelihood does not reproduce the Mg abundance well.

Summary Comparing the distribution of the different likelihoods with the reference values we used to create the spectra gives a good overview of the capabilities of each likelihood. Out of the four candidates, the basic χ^2 ('default') and the subtraction of a continuum ('poly') perform best. In fig. 3.18 and fig. 3.19 we show the full PDF for these likelihoods. Although these likelihoods performed best, they are not perfect and some parameters were not found correctly, namely the Titanium, Magnesium and Chromium abundances and for the 'poly'-likelihood Nickel and Cobalt. We attribute this to the fact, that these elements do not have dominant atomic lines in the spectral region we use for the comparison.

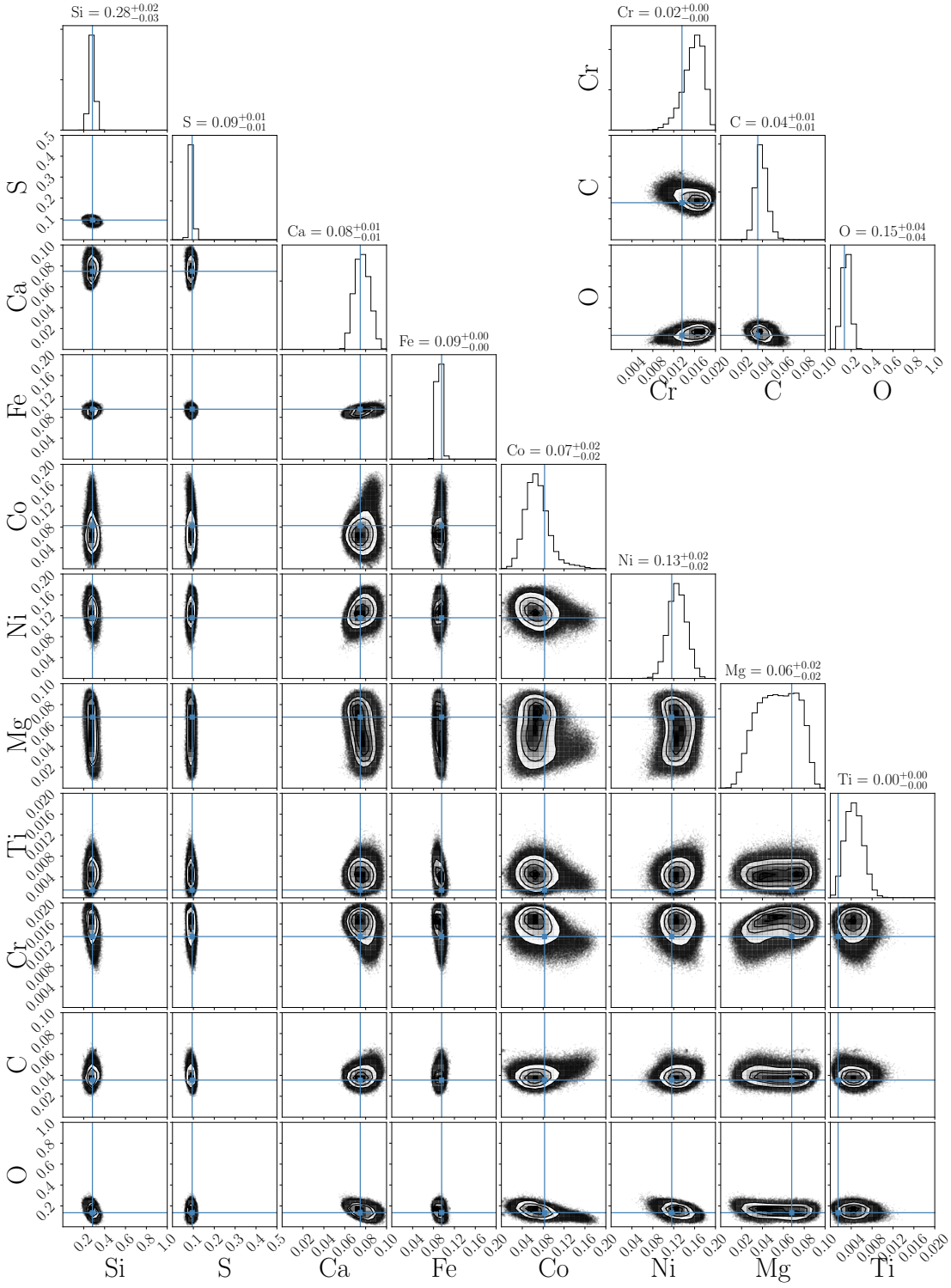

Figure 3.18: Projection of the PDF onto all possible parameter combinations to visualize the distribution of the basic χ^2. The parameters are from top to bottom (and left to right): Si, S, Ca, Fe, Co, Ni, Mg, Ti, Cr, C, O. The blue lines indicate the corresponding parameters of the model used to create the observed spectrum. The limits of all axes coincide with the priors defined in table 3.4 for the logarithmic cube.

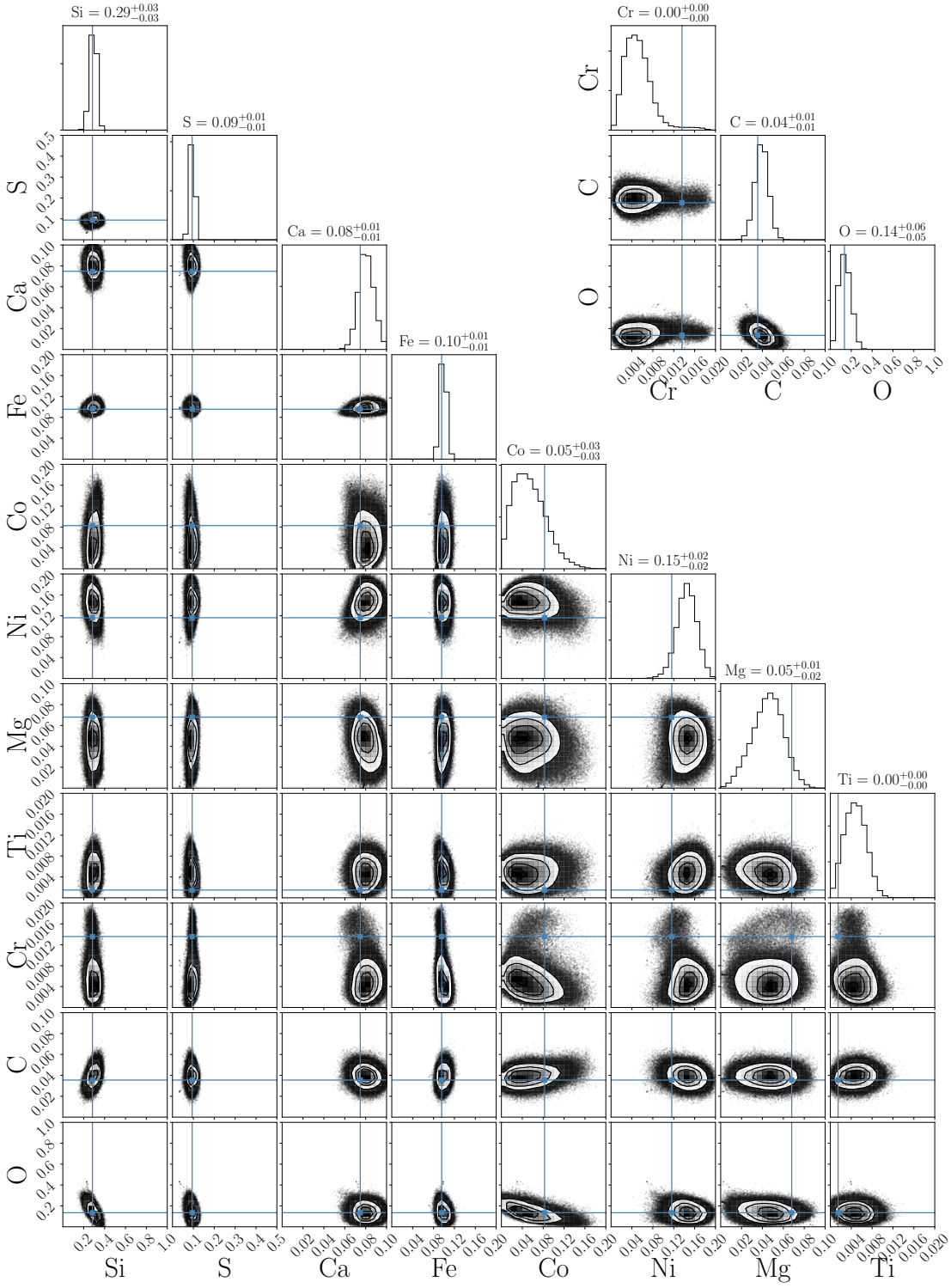

Figure 3.19: Same as fig. 3.18 but with the 'poly'-likelihood. Overall most parameters are well determined, except for Ni, Mg, Ti and Cr.

3.4 Fitting SN 2002bo

We apply the methods (see chapter 2) developed in this thesis to fit the observation of SN 2002bo. We choose this SN because it has previously been extensively studied with abundance tomography methods (see Stehle et al., 2005; Hachinger, 2011).

SN 2002bo was discovered on 9th March 2002 and is located in NGC 3190 with a distance modulus ($\mu = 31.67$; Benetti et al., 2004). The spectroscopic coverage for this supernova starts at about 13 d before maximum light and extends until 368 d which is in the nebular phase. The supernova was the subject of prior studies by Stehle et al. (2005). The supernova was classified as a normal SN Ia but there are some peculiarities concerning the line velocities (see Cacella et al., 2002; Benetti et al., 2004). Past analysis' of the SN found an abnormal amount of Magnesium in the ejecta with abundances up to 30 % (see Stehle et al., 2005).

In a first approach, we use the method developed by Jancauskas et al, in prep. to find the globally best combination of model parameters and compare these to the results obtained by Stehle et al. (2005). This step is not required for the exploration of the parameter space, however with the use of the emulator, repeating the differential evolution approach becomes computationally cheap and provides another set of best-fit model parameters that can be compared with literature. For the generation of spectra we use the emulator trained with the grid we analyzed in section 3.2.2. We compare these spectra with the spectrum of SN 2002bo for the epoch 9.1 d before maximum light, because at this time the systematic uncertainties in the spectra of the grid which were created by TARDIS are assumed to be relatively low.

After obtaining the best fitting combinations of parameters, we start the exploration of the parameter space in a similar manner as outlined in section 3.1 but with the emulator as the source for the spectra and the observation of SN 2002bo as the target. We again use the MCMC algorithm (see section 2.4.2 for details) to generate samples approximate the posterior PDF. We also briefly test a different method, nested sampling (Skilling, 2006) that works similar to MCMC but uses different algorithms to estimate the topology of the whole parameter space.

3.4.1 Differential evolution

In this section, we present our results of using global optimization with the differential evolution algorithm (see Jancauskas et al, in prep.) to find the best fitting set of model parameters for SN 2002bo. We compare them to previous studies by Stehle et al. (2005) and Hachinger (2011). On the one hand we want to assess if the framework functions properly and on the other hand we aim to compare the physical results with literature values.

For the comparison we use the likelihoods defined in section 3.3 and we use the spectral emulator (see section 2.3) trained with the grid used in section 3.2.2 relying on the default

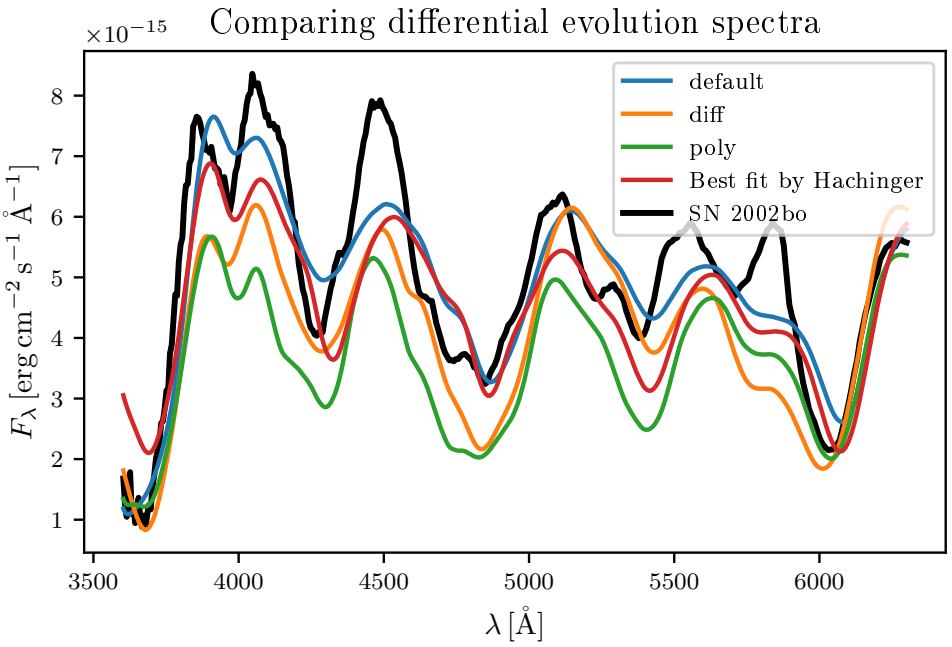

Figure 3.20: Plot showing emulated spectra for the best fit by Hachinger (2011) (red) and the best fit obtained using differential evolution for different likelihoods in comparison to the observation of SN 2002bo (black) 9.1 d before maximum.

parameters to generate the spectra. We use 100 PCs and a random subset of 1000 out of the 36 500 available points as the grid for the interpolation.

In fig. 3.20 we compare the observation with a TARDIS spectrum for the values found by (Hachinger, 2011, henceforth SH fit) and the result of using the differential evolution algorithm with three of the likelihoods presented in section 3.3. We can see, that the 'diff' and 'poly'-likelihoods do not fit the observation. The most probable reason is that these likelihoods do not account for a continuum. As a result, their line profiles match the observation but there is a constant offset and thus, these likelihoods need adjustments to be viable. However, assessing whether the SH fit or our results with the 'default' likelihood match the observation better, is a challenge. The χ^2 does not fit the absorption line at 4000 Å, the peak at 4100 Å and the absorption at 4300 Å but it matches the spectrum better above 5000 Å. One could argue that both, the SH fit and the differential evolution result from the 'default' likelihood are equally good, however if we take the importance of selected lines into account, the fit by Hachinger (2011) performs better.

To get some statistical insight about the results produced by differential evolution, we repeat this experiment and run the algorithm 50 times and compare the results. In fig. 3.21 we visualize the spread of the best fitting spectra and show the mean and

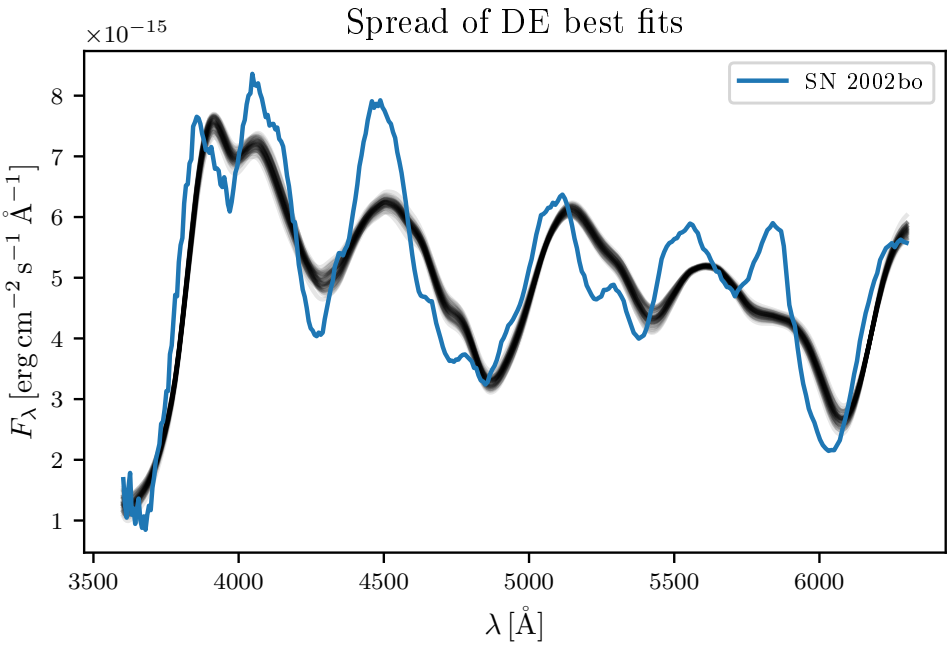

Figure 3.21: Overplotting 50 results obtained by differential evolution (black, $\alpha = 5\,\%$) and the original spectrum of SN 2002bo at 9.1 d before maximum.

standard deviation for the fit results in table 3.7. Some abundances, like Carbon, Cobalt and Oxygen for example have a large standard deviation of approximately 100 % which explains why Jancauskas et al, in prep. found multiple solutions that fit the observation equally well. Therefore it is important to not only find the maximum likelihood point, but to also explore the associated posterior PDF which is the topic of the next section.

Table 3.7: Average value (μ) and standard deviation (σ) of values obtained by running differential evolution repeatedly (50 times) with different starting positions. Large standard deviations (in particular for C, Co and O) highlight the degeneracies in the parameter space and explain why Jancauskas et al, in prep. found multiple solutions. Note that each row contains two abundances.

	'default'		'poly'	
	μ	σ	μ	σ
Si	1.54×10^{-1}	1.54×10^{-2}	4.45×10^{-1}	6.56×10^{-2}
S	2.05×10^{-2}	2.33×10^{-3}	2.40×10^{-1}	7.38×10^{-2}
Ca	7.76×10^{-2}	2.75×10^{-3}	7.11×10^{-2}	3.42×10^{-2}
Fe	1.63×10^{-2}	1.69×10^{-3}	4.56×10^{-2}	8.47×10^{-3}
Co	7.64×10^{-3}	6.00×10^{-3}	1.10×10^{-2}	9.82×10^{-3}
Ni	1.86×10^{-3}	1.48×10^{-3}	4.33×10^{-3}	3.67×10^{-3}
Mg	1.78×10^{-2}	7.66×10^{-3}	8.84×10^{-2}	1.01×10^{-2}
Ti	1.77×10^{-3}	6.50×10^{-4}	9.77×10^{-4}	1.19×10^{-3}
Cr	1.22×10^{-4}	1.03×10^{-4}	1.07×10^{-3}	1.10×10^{-3}
C	2.19×10^{-3}	2.21×10^{-3}	1.43×10^{-2}	1.35×10^{-2}
O	7.00×10^{-1}	9.60×10^{-1}	7.74×10^{-2}	7.79×10^{-1}

3.4.2 Markov chain Monte Carlo

Here we present the results of combining the tools we developed and using them to explore the parameter space of SN 2002bo. For that reason, we repeat our approach from section 3.1 but use the spectral emulator from the previous section (section 3.4.1) to generate the spectra required by the likelihood. We use the EMCEE (see section 2.4.2) algorithm to explore the parameter space with two different likelihoods, the basic χ^2 ('default') and the χ^2 with a subtracted continuum ('poly'). These likelihoods are covered in detail in section 3.3.

We use EMCEE with 300 walkers and run it for 10^5 iterations. The original plan, to randomly distribute the initial points for the walkers and let them converge to the area with the maximum was not successful, because we noticed that the chains would converge to multiple distinct areas with no mixing between them (see fig. 3.22). This is further evidence for the degeneracy of the parameter space observed by Jancauskas et al, in prep. and the high standard deviation in the results obtained through differential evolution. In order to force the walkers to explore one peak, we initialized them to a small area surrounding the best fit of Hachinger (2011). This approach yields acceptable results, which means the walkers converged and the Markov chains we obtained had an autocorrelation of approximately 100 samples. That means, after filtering the burn-in (10^4 samples)

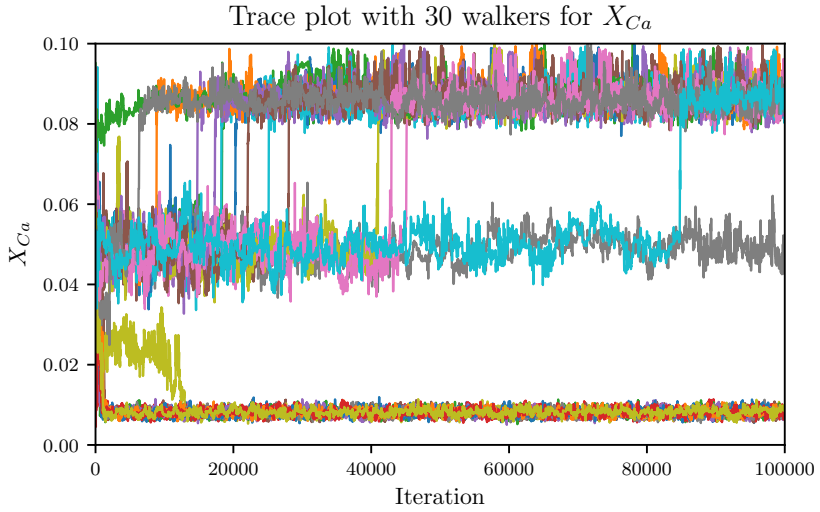

Figure 3.22: We plot the trace of 30 chains after running EMCEE with the 'poly'-likelihood with randomly distributed starting positions. This plot shows well how the chains converged to three different areas with high probability. This is not the desired outcome, instead the goal is for all chains to converge to the same area and explore that. Note that this is not the data used for fig. 3.25.

and correlated samples, we are left with 2.7×10^5 independent points representing the parameter space for each likelihood.

Before we analyze the parameter space in detail, we look at the spectra that dominate the area surrounding the peak (see fig. 3.23). Comparing our result with the spectra obtained using differential evolution (see fig. 3.21), we observe that the spectra for the 'default' likelihood match the observation of SN 2002bo well, whereas the 'poly'-likelihood has the same problem as with differential evolution: The continuum is not taken into account for the comparison and thus, the spectra do not match. Therefore we will focus the discussion on the 'default'-likelihood.

In fig. 3.24 ('default') and fig. 3.25 ('poly') we show the topology associated with the likelihoods. We compare the contours we obtain with the maximum likelihood points found by Stehle et al. (2005); Hachinger (2011) and find overall good agreement. We compare the maximum likelihood model parameters for SN 2002bo found by the previous studies with our points of maximum likelihood in table 3.8. Only the Magnesium and Oxygen abundances do not agree with literature values. This is most likely caused by the limitations of the priors for the Magnesium abundance, which has been reported in literature to be unusually high for SN 2002bo (see Stehle et al., 2005). The inability of the algorithm to fit Magnesium correctly effected the Oxygen abundance which had to make up for the missing Magnesium.

It is important to note that by initializing the Markov chains to the area surrounding the SH fit, we constrained the area EMCEE would explore as the ability of the algorithm to descent into low likelihood regions is limited. Thus, we did not explore the two other peaks for Calcium which are visible in fig. 3.22. While it might be possible to explore the whole parameter space with EMCEE by tweaking its parameters, there are other tools, like Nestle (see Barbary, 2014) that are designed for this task.

In summary, we show that the EMCEE algorithm can be used together with the spectral emulator to explore the parameter space of a real supernova. We find the main problem of this approach to be the likelihood function which is not yet good enough in determining the quality of a fit. In the case of comparing a real supernova with a synthetic spectrum, it is important to quantify the systematic uncertainties of the radiative transfer code in some way as they dominate the difference between model and observation in some wavelength regions.

Table 3.8: A list of literature abundances for SN 2002bo found by Hachinger (2011) (SH fit) and Stehle et al. (2005) (Stehle) in comparison to the values we obtain using the EMCEE and NESTLE methods with different likelihood functions. Note that our calculations were done with $v_\mathrm{i} = 11\,700\,\mathrm{km\,s^{-1}}$ and $T_\mathrm{i} = 10\,000\,\mathrm{K}$, the same values used in Hachinger (2011). However they are different from the values used by Stehle et al. (2005) ($v_\mathrm{i} = 13\,900\,\mathrm{km\,s^{-1}}$ and $T_\mathrm{i} = 9239\,\mathrm{K}$).

	SH fit	Stehle	EMCEE		NESTLE	
			'default'	'poly'	'default'	'poly'
Si	2.5×10^{-1}	3.1×10^{-1}	2.1×10^{-1}	5.0×10^{-1}	1.2×10^{-1}	3.0×10^{-1}
S	4.5×10^{-2}	$6\ \times10^{-2}$	1.7×10^{-2}	1.3×10^{-1}	2.5×10^{-2}	2.3×10^{-5}
Ca	1.0×10^{-2}	$2\ \times10^{-2}$	1.5×10^{-2}	9.7×10^{-3}	5.7×10^{-2}	8.9×10^{-3}
Fe	8.0×10^{-3}	$3\ \times10^{-2}$	1.5×10^{-2}	3.9×10^{-2}	1.8×10^{-2}	1.2×10^{-1}
Co	9.9×10^{-3}	-	2.2×10^{-3}	3.6×10^{-3}	1.8×10^{-2}	1.9×10^{-2}
Ni	4.6×10^{-3}	1.1×10^{-1}	1.4×10^{-3}	9.5×10^{-3}	6.7×10^{-5}	6.6×10^{-10}
Mg	1.0×10^{-1}	2.8×10^{-1}	1.6×10^{-2}	9.7×10^{-2}	1.8×10^{-2}	7.9×10^{-2}
Ti	1.0×10^{-3}	$6\ \times10^{-3}$	4.0×10^{-3}	1.3×10^{-4}	1.6×10^{-3}	2.5×10^{-10}
Cr	7.0×10^{-4}	$6\ \times10^{-3}$	2.1×10^{-4}	1.9×10^{-4}	7.0×10^{-10}	2.7×10^{-3}
C	8.0×10^{-4}	-	2.5×10^{-3}	1.0×10^{-2}	9.0×10^{-10}	8.5×10^{-3}
O	5.5×10^{-1}	1.9×10^{-1}	7.1×10^{-1}	2.0×10^{-1}	7.5×10^{-1}	4.6×10^{-1}

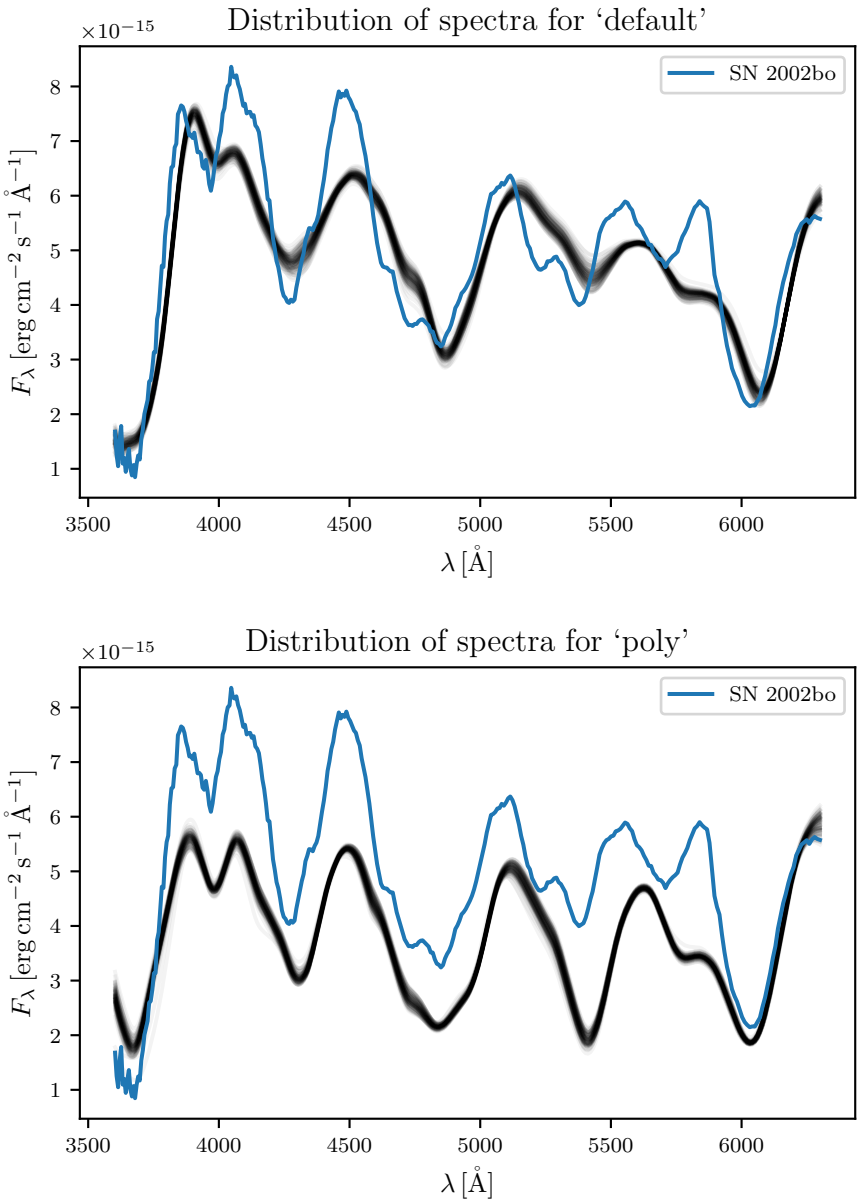

Figure 3.23: Plot showing the spread in the spectra near the maximum probability point. This was done for the basic χ^2(top) and the 'poly'-likelihood (bottom). The plot includes the spectrum of SN 2002bo(blue) and emulated spectra (black, $\alpha = 5\,\%$) for a subset of 100 points from the samples generated by EMCEE.

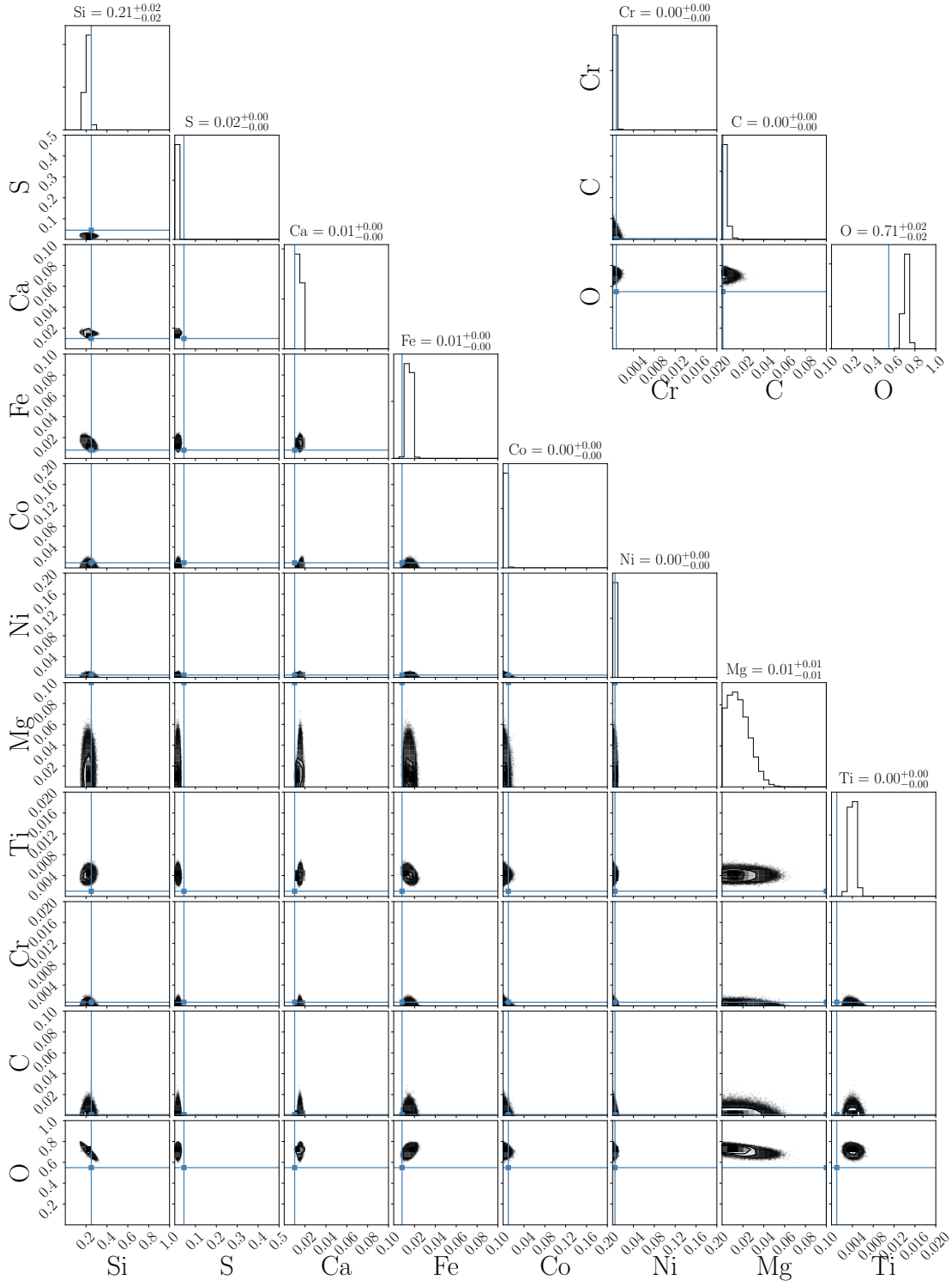

Figure 3.24: Same as fig. 3.18 but we compare the emulator spectra against SN 2002bo and use the 'default' likelihood as metric for comparison. Again, the parameters from top to bottom (and left to right) are Si, S, Ca, Fe, Co, Ni, Mg, Ti, Cr, C and O. The blue lines mark the abundances of the fit by Hachinger (2011). As before, the blue lines represent the values of the fit by Hachinger (2011). The contours represent $1\,\sigma$ confidence intervals.

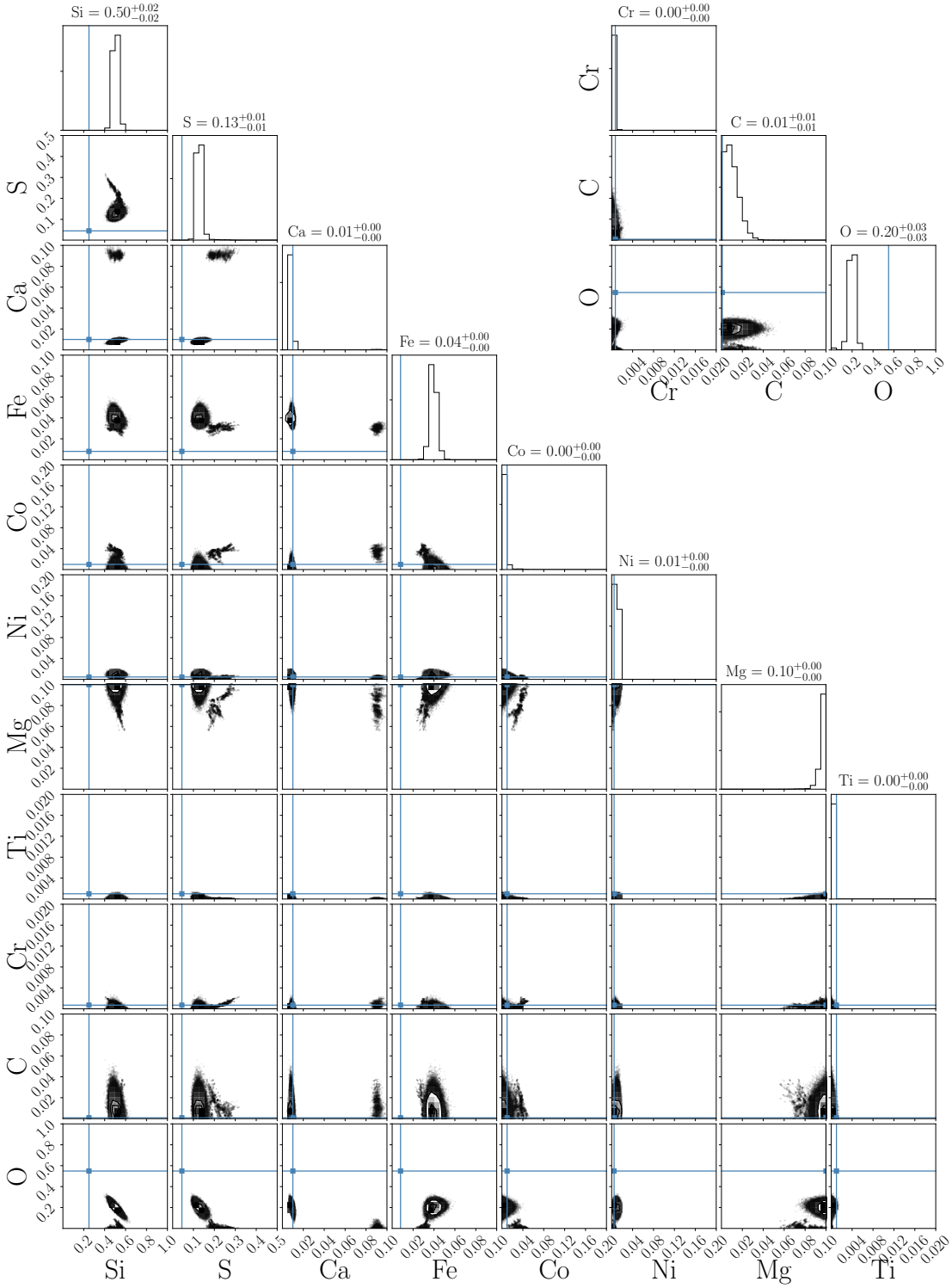

Figure 3.25: Same as fig. 3.24 but we use the 'poly'-likelihood. Again, the parameters from top to bottom (and left to right) are Si, S, Ca, Fe, Co, Ni, Mg, Ti, Cr, C and O. As before, the blue lines represent the values of the fit by Hachinger (2011). The contours represent $1\,\sigma$ confidence intervals.

3.4.3 Nested sampling

Lastly, we test Nestle (see Barbary, 2014), an implementation of the MultiNest algorithm (see Feroz et al., 2009) which is a different method to explore the parameter space. The approach is similar to EMCEE, however, the key difference is that NESTLE tries to sample the whole PDF space defined by the priors. This is the reason we try this algorihm, because it allows us to explore all parts of the PDF and not only the area around the maximum likelihood point. We run NESTLE with the two likelihood functions 'default' and 'diff' to compare spectra generated with the emulator from section 3.4.2 with the observation of SN 2002bo.

In contrast to the MCMC algorithm where the run-time is fixed by the number of iterations, the execution of this test is dynamic and dependent on the topology the parameter space that is explored, which took an unexpected amount of computation time. While these tests with the basic χ^2 took approximately a day to finish, the analysis with the 'diff'-likelihood, which uses the derivative of the spectrum, ran for over a month. Unfortunately the test which used the χ^2 with a subtracted polynomial had to be stopped after running for two months without results. The increase in runtime is due to the increased complexity of the topology that is associated with these likelihoods.

We start presenting the results by showing a comparison of the best fitting spectra in fig. 3.26. As can be seen, the 'diff' likelihood performs not well and has similar problems as the 'poly' likelihood which we already discussed above (see section 3.4.2). Therefore we will focus the analysis on the 'default'-likelihood which performs equally well as the

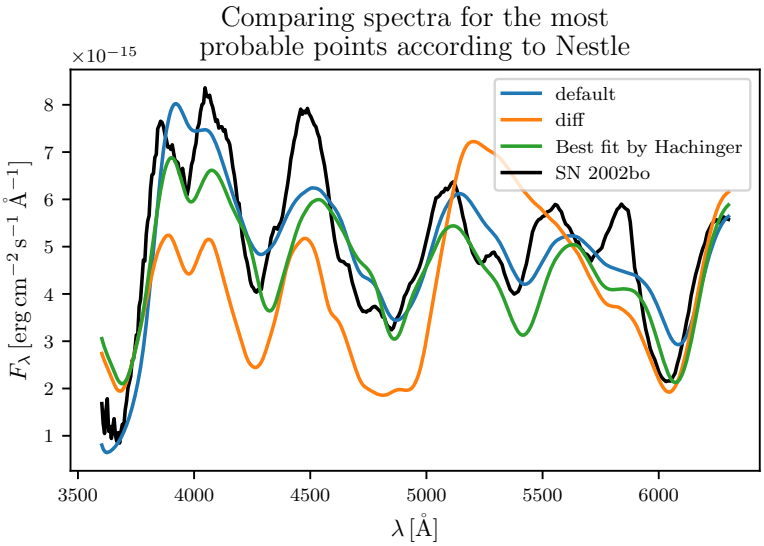

Figure 3.26: The plot shows the observation of SN 2002bo (black) in comparison to emulator spectra for the 'default' (dark blue) and 'diff' (yellow) likelihood as well as the SH fit (green). The 'diff'-likelihood is clearly an outlier as the spectra do not match.

fit by Hachinger (2011). The only major concern in this spectrum is that the Silicon line at approximately 6000 Å does not match well.

For the comparison to Hachinger (2011) and Stehle et al. (2005), we will focus on the 'default' likelihood fits as this seems to find better matches to the observations as opposed to the 'poly' likelihood. In fig. 3.27 we show the topology associated with the 'default' likelihood and, for completeness, we also show the same plot for the 'poly' likelihood (see fig. 3.28). We list the parameters for our maximum likelihood points obtained by nestle together with parameters found by the previous studies and our results from EMCEE in table 3.8. Overall the confidence contours seem to agree with both previous studies. In addition to the agreement, we, however, also show the space is degenerate with multiple solutions that are equally good but not presented in the literature. While most elements are broadly consistent with Stehle et al. (2005); Hachinger (2011), Magnesium and Calcium show noticeable differences.

For Magnesium this is likely due to the fact that we allow for a normal Magnesium abundance but SN 2002bo has been shown to be especially abundant in this element (up to $X_{\mathrm{Mg}} = 0.3$; Stehle et al., 2005). Calcium is seemingly multivariate where Hachinger et al. (2009) and Stehle et al. (2005) have focused on one part of the PDF each while the NESTLE algorithm finds all three. As mentioned in section 3.4.2, the EMCEE algorithm was also able to find multiple peaks when initialized randomly, however this had a negative impact on the performance of the algorithm and we chose to sample from one peak instead. This again emphasizes the point that it is crucial to explore the parameter space and not only present maximum likelihood values.

The first exploration of the PDF for these spectra has already shown that there are undocumented degeneracies which need to be considered when comparing these fitted values with theoretical nucleosynthesis models. Our nestle exploration has shown that our method is performing well and can be used in the future to explore this space in more detail.

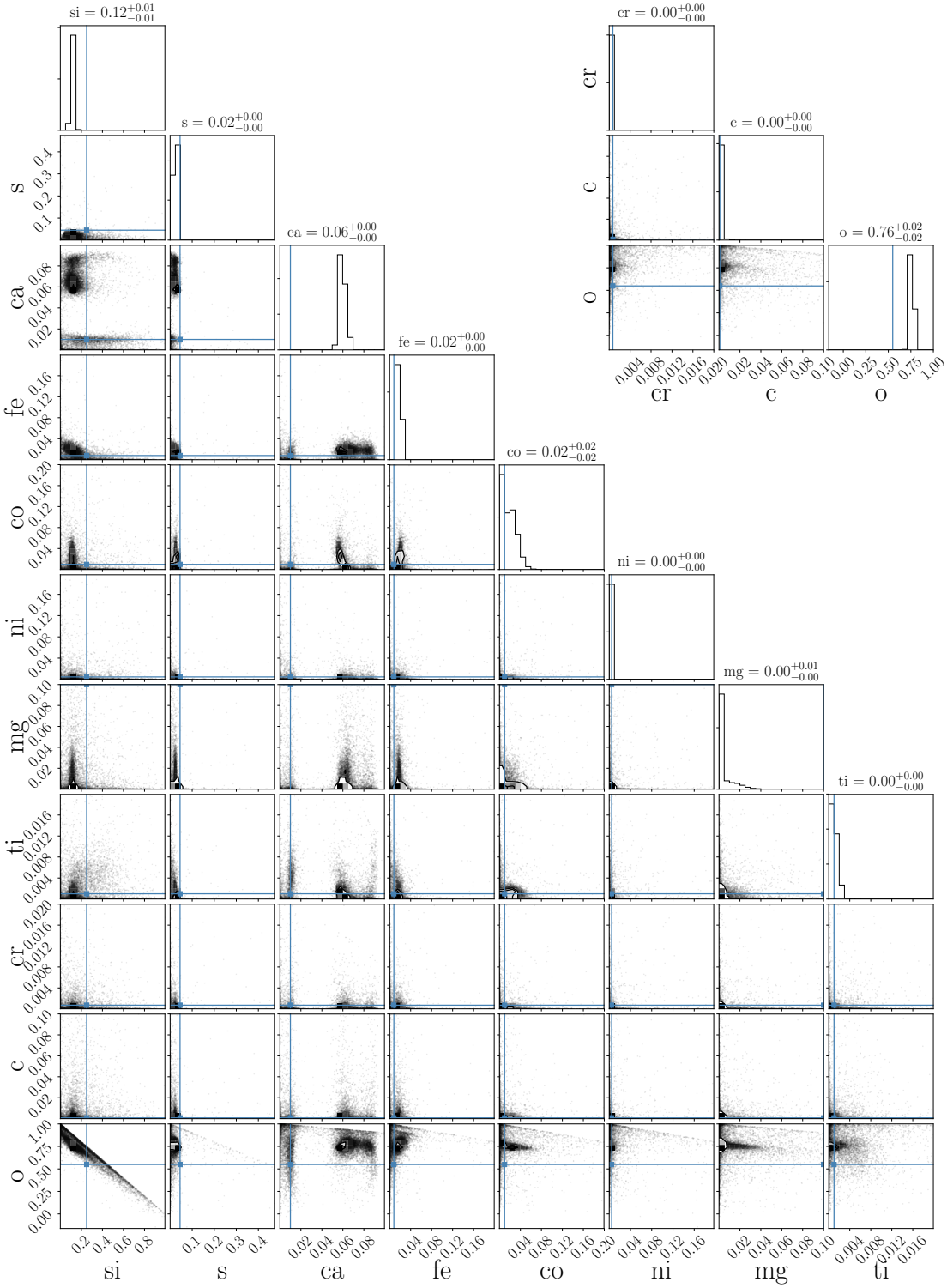

Figure 3.27: Topology of the parameter space for SN 2002bo obtained using NESTLE and the basic χ^2 likelihood. Again, the parameters from top to bottom (and left to right) are Si, S, Ca, Fe, Co, Ni, Mg, Ti, Cr, C and O. As before, the blue lines represent the values of the fit by Hachinger (2011). The contours represent $1\,\sigma$ confidence intervals.

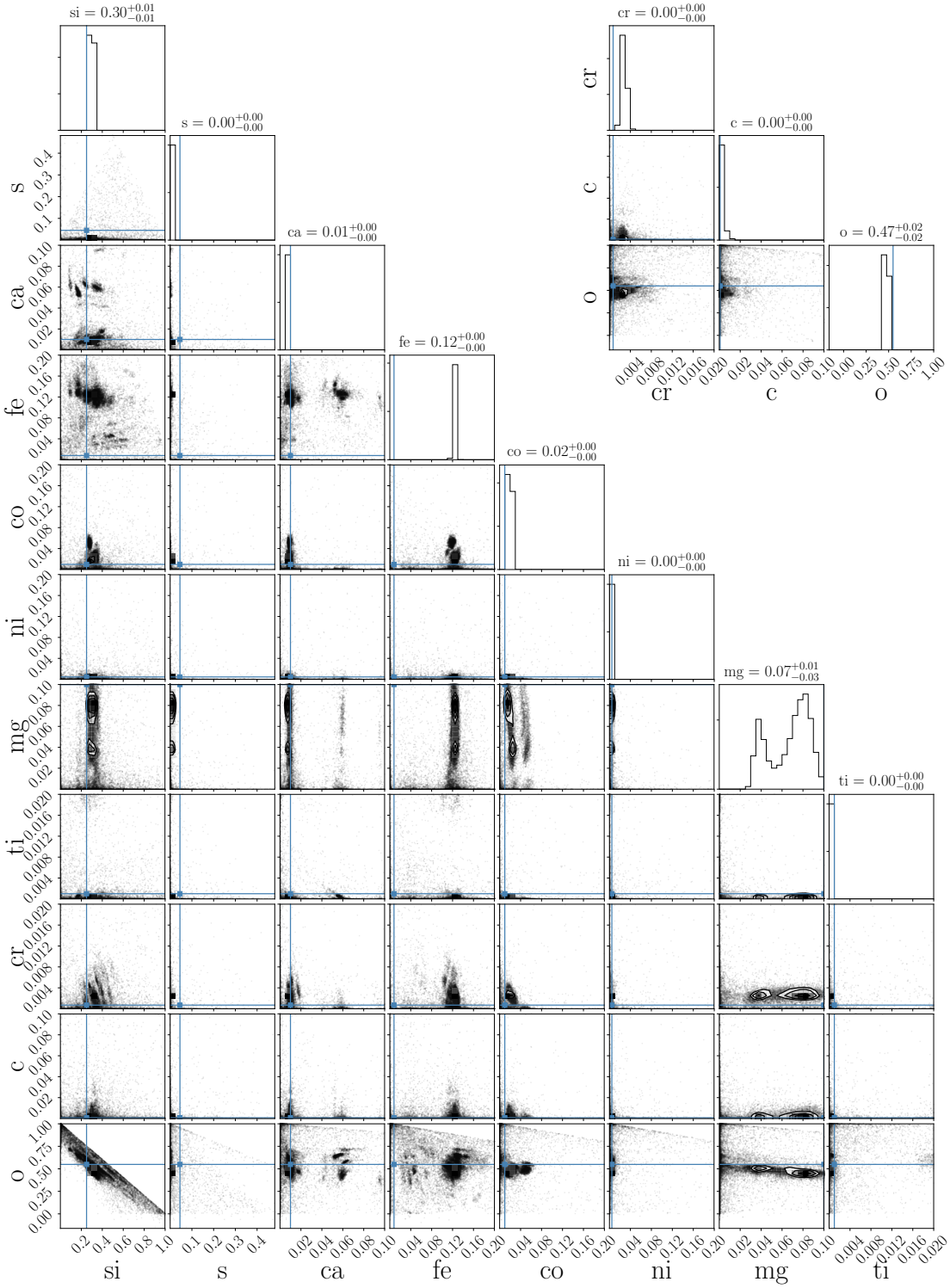

Figure 3.28: Same as fig. 3.27 but we use the 'diff' likelihood. Again, the parameters from top to bottom (and left to right) are Si, S, Ca, Fe, Co, Ni, Mg, Ti, Cr, C and O. As before, the blue lines represent the values of the fit by Hachinger (2011). The contours represent $1\,\sigma$ confidence intervals.

4 Conclusions

4.1 Summary

In this work we have presented first steps towards automating the abundance tomography method, that is to fit an observed spectral time series of a SN Ia and infer important characteristics and properties of the exploding object.

An important aspect of this approach is the possibility to study the full parameter space with this technique. This, in principle allows us to identify degeneracies and exclude regions parameter space which are clearly incompatible with observations. We use ML techniques in combination with a fast spectral synthesis code (TARDIS) to achieve this automation. However, our first tests showed, that even fast radiative transfer codes are orders of magnitude too slow to efficiently explore the parameter space.

We therefore develop a spectral emulator, which is a means to interpolate efficiently in a precomputed grid of synthetic data and at the same time provide information about the uncertainties associated with the interpolation. In a series of tests we analyze the performance and accuracy of the emulator. In particular we validate that the results from the emulator are compatible with those explicitly calculated with TARDIS. This spectral emulator performs well if the abundances in the ejecta are the only variable parameters of the analysis. However, so far we were not able to successfully incorporate T_i and v_i as additional model parameters due to their non-linear effect on the formation of the spectra.

Automating abundance tomography also requires a way to asses the quality of a fit. We use Bayesian Statistics to combine prior knowledge about the parameter space (*e.g.,*nucleosynthesis constraints on the ratios of certain elements) and a likelihood function to formulate a posterior probability. For the likelihood function we test different approaches and settle for a basic χ^2 to comparing the spectra. With this likelihood we can converge on the correct abundances for most elements reasonable well. Elements that do not fit well have no dominant lines in the spectral window we consider for the fit and thus the poor fit is expected.

In a final application of our framework we apply our analysis to actual observations, by studying SN 2002bo with the techniques developed during this thesis. In particular we were able to derive abundances, similar to those reported in literature. Parameters whose literature values were not well reproduced lie close to the boundaries of the priors

we imposed (*e.g.*,because of nucleosynthesis calculations). Thus, these deviations are not unexpected.

Based on the work presented here, we conclude, that automating the complicated fitting process of Type Ia spectra is a promising technique in the future analysis of SN Ia. However, the framework presented here is only a starting point and has to be extended to reach its full potential.

4.2 Outlook

So far we only looked at a uniform abundance model for the spectrum for one epoch. But our goal is to have a stratified abundance profile and fit a time series of spectra. However, this may pose problems for our approach. Our current simple model has ten parameters, with five epochs we would already increase that to 60 parameters. Our current approach cannot handle these many dimensions because we reach the limits of what is computationally feasible (both with respect to time and hardware requirements).

To overcome these challenges, we propose to explore two possible strategies. One possible approach would be to train the emulator iteratively. That means, instead of precomputing a grid of spectra, we start with a small number of training points and decide based on the uncertainty of the interpolation whether the interpolation results will be used or if a synthetic spectrum is explicitly created with TARDIS. This would then be included in the training data of the emulator. The benefit would be that only for points needed in the exploration of the parameter space TARDIS calculations would be performed. However this approach would not allow the generation of a precomputed grid in a parallelized way, thus resulting in long serial run times of the code.

A completely different approach that would scale well with many parameters would be to interpolate the likelihood directly instead of the spectrum. The benefit would be a significant reduction in computational resources. This approach could be combined with the previously suggested iterative training of the emulator to create a tool that determines whether or not the prediction is accurate enough and then update itself with the new information. A combination of the options above would be to use an emulator for the likelihood to limit the parameter space down to a small region in which the spectral emulator could actually work if the grid is dense enough. However the approach of emulating the likelihood function requires an accurate definition thereof.

Independent to the abundance tomography method, the field of SN Ia research could benefit in general from an improved likelihood to compare spectra. There are several ideas that could be explored, such as using local GP kernels to deal with model imperfections (see Czekala et al., 2015) or using a metric space for supernova (see Sasdelli et al., 2014) to formulate a likelihood.

With the recent advancements in machine learning and more powerful tools becoming readily available in different disciplines, now is the perfect time to exploit them in astrophysics and abundance tomography in particular.

Acknowledgments

First, I would like to thank Wolfgang Hillebrandt for supervising me and giving me the opportunity to explore science. He reignited my interest for astronomy, the original reason I started to study physics, after having spent most of my studies in the field of condensed matter physics.

I thank Wolfgang Kerzendorf as the head of this project for guiding and assisting me and his ability to explain complex concepts in an easily understandable manner. He taught me to 'not let good be the enemy of perfect', which stopped me over-thinking problems.

Furthermore, I have to thank Ulrich Nöbauer for helping me with the theory and for his constructive feedback on my unfinished versions of the thesis. Then I would like to thank all people at the MPA for providing such an enjoyable environment to work in, especially Christian Vogl and Andreas Flörs for the numerous discussions about physics and the problems of the world.

For his help in questions regarding statistics and MCMC, I want to thank Frederik Beaujean. In a similar manner I want to thank Stefan Taubenberger for showing me the world of observations and Stephan Hachinger for explaining the abundance tomography method in detail to me. I thank Stuart Sim for his guidance in the early stages of this project

As I think Open-Source is an important concept I want to thank all contributors and companies that support it. A special thanks goes to Google and the organizers of Google Summer of Code for allowing me to mentor students working on the TARDIS code during the thesis. For the PYTHON tools they provide I want to thank all people involved in providing libraries, especially NUMPY, SCIPY, MATPLOTLIB and SCIKIT-LEARN.

I want to thank my parents, Karin and Klaus for bringing me up the way they did and encouraging me to explore the world and to always be curios. And I want to thank all my friends and everyone else I forgot to mention.

Lastly I want to thank my girlfriend Tamina for supporting me throughout my studies and especially in the last weeks of my thesis, where she always tried to keep me in a positive mood.

Bibliography

Abbott, D. C. & Lucy, L. B. 1985, ApJ, 288, 679

Ashall, C., Mazzali, P., Bersier, D., Hachinger, S., Phillips, M., Percival, S., James, P., & Maguire, K. 2014, MNRAS, 445, 4427

Barbary, K. 2014, Nestle, GitHub [LINK]

Benetti, S., Meikle, P., Stehle, M., Altavilla, G., Desidera, S., Folatelli, G., Goobar, A., Mattila, S., Mendez, J., Navasardyan, H., Pastorello, A., Patat, F., Riello, M., Ruiz-Lapuente, P., Tsvetkov, D., Turatto, M., Mazzali, P., & Hillebrandt, W. 2004, MNRAS, 348, 261

Branch, D., Doggett, J. B., Nomoto, K., & Thielemann, F.-K. 1985, The Astrophysical Journal, 294, 619

Breiman, L. 2001, Machine Learning, 45, 5 [LINK]

Cacella, P., Hirose, Y., Nakano, S., Kushida, Y., Kushida, R., & Li, W. D. 2002, IAU Circ., 7847

Chakraborty, U. 2008, Advances in Differential Evolution (Springer-Verlag GmbH) [LINK]

Chandrasekhar, S. 1931, ApJ, 74, 81

Colgate, S. A. & McKee, C. 1969, ApJ, 157, 623

Cressie, N. 2015, Statistics for Spatial Data, Wiley Series in Probability and Statistics (Wiley) [LINK]

Czekala, I., Andrews, S. M., Mandel, K. S., Hogg, D. W., & Green, G. M. 2015, The Astrophysical Journal, 812, 128 [LINK]

Do, T., Kerzendorf, W., Winsor, N., Støstad, M., Morris, M. R., Lu, J. R., & Ghez, A. M. 2015, ApJ, 809, 143

Feroz, F., Hobson, M. P., & Bridges, M. 2009, MNRAS, 398, 1601

Foreman-Mackey, D., Hogg, D. W., Lang, D., & Goodman, J. 2013, Publications of the Astronomical Society of the Pacific, 125, 306

Gilfanov, M. & Bogdán, Á. 2010, Nature, 463, 924

Goodman, J. & Weare, J. 2010, Communications in Applied Mathematics and Computational Science, 5, 65

Hachinger, S. 2011, PhD thesis, Technische Universität München, München, Germany

Hachinger, S., Mazzali, P. A., Taubenberger, S., Pakmor, R., & Hillebrandt, W. 2009, MNRAS, 399, 1238

Hastings, W. K. 1970, Biometrika, 57, 97

Heald, M. A. 2003, American Journal of Physics, 71, 1322

Hillebrandt, W., Kromer, M., Röpke, F. K., & Ruiter, A. J. 2013, Frontiers of Physics, 8, 116

Hillebrandt, W. & Niemeyer, J. C. 2000, Annual Review of Astronomy and Astrophysics, 38, 191

Hogg, D. W., Bovy, J., & Lang, D. 2010, ArXiv e-prints

Hoyle, F. & Fowler, W. A. 1960, ApJ, 132, 565

Husser, T.-O., von Berg, S. W., Dreizler, S., Homeier, D., Reiners, A., Barman, T., & Hauschildt, P. H. 2013, Astronomy & Astrophysics, 553, A6

Iben, Jr., I., Nomoto, K., Tornambe, A., & Tutukov, A. V. 1987, ApJ, 317, 717

Iben, Jr., I. & Tutukov, A. V. 1984, ApJS, 54, 335

Jeffreys, S. H. 2011, Scientific Inference (Cambridge University Press) [LINK]

Jolliffe, I. T. 2002, Principal Component Analysis (Springer-Verlag GmbH) [LINK]

Kasen, D., Thomas, R. C., & Nugent, P. 2006, ApJ, 651, 366

Kerzendorf, W. E. & Sim, S. A. 2014, Monthly Notices of the Royal Astronomical Society, 440, 387

Khokhlov, A. M. 1991, A&A, 245, 114

Kippenhahn, R., Weigert, A., & Weiss, A. 2012, Stellar Structure and Evolution (Springer Berlin Heidelberg) [LINK]

Kobayashi, C., Tsujimoto, T., Nomoto, K., Hachisu, I., & Kato, M. 1998, ApJ, 503, L155

Kromer, M. & Sim, S. A. 2009, MNRAS, 398, 1809

Kuchner, M. J., Kirshner, R. P., Pinto, P. A., & Leibundgut, B. 1994, ApJ, 426, 89

Lamers, H. J. G. L. M. & Cassinelli, J. P. 2015, Introduction to Stellar Winds (CAMBRIDGE UNIV PR) [LINK]

Leibundgut, B. 2000, Astronomy and Astrophysics Review, 10, 179

Li, W. D., Filippenko, A. V., Treffers, R. R., Friedman, A., Halderson, E., Johnson, R. A., King, J. Y., Modjaz, M., Papenkova, M., Sato, Y., & Shefler, T. 2000, 522, 103

Lucy, L. B. 1999, A&A, 344, 282

Matteucci, F. & Greggio, L. 1986, A&A, 154, 279

Mazzali, P. A. 2000, A&A, 363, 705

Mazzali, P. A. & Lucy, L. B. 1993, A&A, 279, 447

Mazzali, P. A., Sullivan, M., Filippenko, A. V., Garnavich, P. M., Clubb, K. I., Maguire, K., Pan, Y.-C., Shappee, B., Silverman, J. M., Benetti, S., Hachinger, S., Nomoto, K., & Pian, E. 2015, MNRAS, 450, 2631

Mihalas, D. 1978, Stellar Atmospheres (A Series of books in astronomy and astrophysics) (W H Freeman & Co (Sd)) [LINK]

Nocedal, J. & Wright, S. 2006, Numerical Optimization (Springer-Verlag GmbH) [LINK]

Nomoto, K. 1982, ApJ, 253, 798

Nomoto, K., Thielemann, F.-K., & Yokoi, K. 1984, ApJ, 286, 644

Olling, R. P., Mushotzky, R., Shaya, E. J., Rest, A., Garnavich, P. M., Tucker, B. E., Kasen, D., Margheim, S., & Filippenko, A. V. 2015, Nature, 521, 332

Pakmor, R., Kromer, M., Röpke, F. K., Sim, S. A., Ruiter, A. J., & Hillebrandt, W. 2010, Nature, 463, 61

Pakmor, R., Kromer, M., Taubenberger, S., Sim, S. A., Röpke, F. K., & Hillebrandt, W. 2012, ApJ, 747, L10

Pankey, Jr., T. 1962, PhD thesis, HOWARD UNIVERSITY.

Perlmutter, S., Aldering, G., Goldhaber, G., Knop, R. A., Nugent, P., Castro, P. G., Deustua, S., Fabbro, S., Goobar, A., Groom, D. E., Hook, I. M., Kim, A. G., Kim, M. Y., Lee, J. C., Nunes, N. J., Pain, R., Pennypacker, C. R., Quimby, R., Lidman, C., Ellis, R. S., Irwin, M., McMahon, R. G., Ruiz-Lapuente, P., Walton, N., Schaefer, B., Boyle, B. J., Filippenko, A. V., Matheson, T., Fruchter, A. S., Panagia, N., Newberg, H. J. M., Couch, W. J., & Project, T. S. C. 1999, ApJ, 517, 565

Phillips, M. M. 1993, ApJ, 413, L105

Pskovskii, Y. P. 1984, Soviet Ast., 28, 658

Rasmussen, C. & Williams, C. 2006, Gaussian Processes for Machine Learning, Adaptative computation and machine learning series (MIT University Press Group Ltd) [LINK]

Reinecke, M., Hillebrandt, W., & Niemeyer, J. C. 2002, A&A, 386, 936

Riess, A. G., Filippenko, A. V., Challis, P., Clocchiatti, A., Diercks, A., Garnavich, P. M., Gilliland, R. L., Hogan, C. J., Jha, S., Kirshner, R. P., Leibundgut, B., Phillips, M. M., Reiss, D., Schmidt, B. P., Schommer, R. A., Smith, R. C., Spyromilio, J., Stubbs, C., Suntzeff, N. B., & Tonry, J. 1998, AJ, 116, 1009

Sasdelli, M., Hillebrandt, W., Aldering, G., Antilogus, P., Aragon, C., Bailey, S., Baltay, C., Benitez-Herrera, S., Bongard, S., Buton, C., Canto, A., Cellier-Holzem, F., Chen, J., Childress, M., Chotard, N., Copin, Y., Fakhouri, H. K., Feindt, U., Fink, M., Fleury, M., Fouchez, D., Gangler, E., Guy, J., Ishida, E. E. O., Kim, A. G., Kowalski, M., Kromer, M., Lombardo, S., Mazzali, P. A., Nordin, J., Pain, R., Pecontal, E., Pereira, R., Perlmutter, S., Rabinowitz, D., Rigault, M., Runge, K., Saunders, C., Scalzo, R., Smadja, G., Suzuki, N., Tao, C., Taubenberger, S., Thomas, R. C., Tilquin, A., & Weaver, B. A. 2014, Monthly Notices of the Royal Astronomical Society, 447, 1247

Sasdelli, M., Ishida, E. E. O., Vilalta, R., Aguena, M., Busti, V. C., Camacho, H., Trindade, A. M. M., Gieseke, F., de Souza, R. S., Fantaye, Y. T., & Mazzali, P. A. 2016, MNRAS, 461, 2044

Savitzky, A. & Golay, M. J. E. 1964, Analytical Chemistry, 36, 1627

Shapiro, S. L. & Teukolsky, S. A. 1983, Black holes, white dwarfs, and neutron stars: The physics of compact objects (Wiley-VCH) [LINK]

Sivia, D. & Skilling, J. 2006, Data Analysis (Oxford University Press) [LINK]

Skilling, J. 2006, Bayesian Analysis, 1, 833

Sokal, A. D. 1996, Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms, Lecture notes [LINK]

Stehle, M., Mazzali, P. A., Benetti, S., & Hillebrandt, W. 2005, Monthly Notices of the Royal Astronomical Society, 360, 1231

Stein, M. 1987, Technometrics, 29, 143

Thomas, R. C., Nugent, P. E., & Meza, J. C. 2011, PASP, 123, 237

Truran, J. W., Arnett, W. D., & Cameron, A. G. W. 1967, Canadian Journal of Physics, 45, 2315

Wang, B., Liu, D., Jia, S., & Han, Z. 2013, Proceedings of the International Astronomical Union, 9, 442

Webbink, R. F. 1984, ApJ, 277, 355

Whelan, J. & Iben, Jr., I. 1973, ApJ, 186, 1007

Woods, T. E. & Gilfanov, M. 2013, MNRAS, 432, 1640

—. 2014, MNRAS, 439, 2351

Woosley, S. E. & Weaver, T. A. 1986, Annual Review of Astronomy and Astrophysics, 24, 205

Yoon, S.-C., Podsiadlowski, P., & Rosswog, S. 2007, Monthly Notices of the Royal Astronomical Society, 380, 933